VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Využití informačních technologií, umělé inteligence a strojového učení pro provoz moderních budov

Use of Information Technology, Artificial Intelligence and Machine Learning for the Operation of Modern Buildings

Milan Křivánek

Bakalářská práce

Vedoucí práce: doc. Ing. Bohumil Horák, Ph.D

Ostrava, 2021

Abstrakt

Cílem této bakalářské práce je navrhnout řešení monitoringu moderních budov pro jejich možné řízení využívající informační technologie, umělou inteligenci a strojové učení. Realizace bude provedena po technické i programové části. Teoretická část se bude věnovat aktuálním možnostem pro řízení budov a monitorování provozu. V praktické části bude vytvořen ucelený systém pro monitoring a provoz v domě s důrazem na záznam teploty, vlhkosti, tlaku a jejich vizualizaci na vlastním domácím web serveru.

Klíčová slova

Budova; monitoring budov; řízení budov; internet věcí; strojové učení; umělá inteligence; expertní systémy;

Abstract

The aim of this bachelor thesis is to design a monitoring solution for modern buildings for their possible management using information technology, artificial intelligence and machine learning. The implementation will be done technically and programmatically. The theoretical part will focus on current possibilities for building management and their monitoring. In the practical part a comprehensive system will be created for monitoring and operation in the house with an emphasis on recording temperature, humidity, pressure and their visualization on my home web server.

Keywords

Building; monitoring of buildings; management of buildings; internet of things; artificial intelligence; machine learning; expert systems

Poděkování

Rád bych poděkoval svému vedoucímu práce doc. Ing. Bohumilu Horákovi, Ph.D za odbornou pomoc a konzultace při vytváření této bakalářské práce. Dále bych chtěl také poděkovat Ing. Milanu Křivánkovi za cenné rady při získávání informací pro technickou část práce.

Obsah

Seznam použitých zkratek a symbolů

ADC	–	převodník analogového signálu na digitální (analog to digital converter)
CPU	–	centrální procesorová jednotka (central processing unit)
DHCP	–	dynamický hostovací konfigurační protocol (dynamic host configuration protocol)
DDNS	–	dynamický systém doménových jmen (dynamic domain name system)
DNS	–	systém doménových jmen (domain name system)
HTTP	–	hypertextový přenosový protocol (hypertext transfer protocol)
HTML	–	hypertextový značkovací jazyk (hypertext markup language)
IDE	–	integrované vývojové prostředí (integrated development environment)
IoT	–	internet věcí (internet of things)
IP	–	internetový protokol (internet protocol)
ISP	–	poskytovatel internetového připojení (internet service provider)
JSON	–	JavaScriptový objektový zápis (JavaScript object notation)
LED	–	elektroluminiscenční dioda (light-emitting diode)
NAS	–	síťové úložiště (network attached storage)
NAT	–	překlad síťových adres (network address translation)
NTP	–	síťový časový protocol (network time protocol)
OS	–	operační system (operation system)
PLC	–	programovatelný logický automat (programmable logic controller)
RAID	–	vícenásobné pole nezávislých disků (redundant array of independent disks)
RAM	–	paměť s libovolným přístupem (random access memory)
ROM	–	paměť jen pro čtení (read only memory)
RTC	–	hodiny reálného času (real time clock)
ML	–	strojové učení (machine learning)
TCP	–	transportní protokol (transmission control protocol)

UART	–	univerzální asynchronní přijímač a vysílač (universal asynchronous receiver-transmitter)
AI	–	umělá inteligence (artificial intelligence)
URI	–	jednotný identifikátor zdroje (uniform resource identifier)
URL	–	jednotný lokátor zdroje (uniform resource locator)
URN	–	jednotné jméno zdroje (uniform resource name)
USB	–	univerzální sériová sběrnice (universal serial bus)
VPN	–	virtuální privátní síť (virtual private network)
XML	–	rozšiřitelný značkovací jazyk (extensible markup language)
I-P-O	–	vstup, zpracování, výstup (input, processing, output)

Seznam obrázků

Kapitola 1

Úvod

V dnešní moderní době je člověk ze všech stran obklopen chytrými zařízeními, která ulehčují práci, umožňují používat rozličné funkce pro realizaci lidských nápadů a poskytují jistý komfort. Může se jednat např. o chytrý telefon nebo chytrou infrastrukturu měst a budov. Chytrá jsou tato zařízení proto, že dokáží spolu komunikovat, číst data z reálného světa okolo nás a následně podle nich vyhodnocovat závěry. Celý tento proces integrace chytrých zařízení do různých odvětví našeho života odstartovalo vynalezení internetu, který se postupným rozvojem dostal z průmyslového odvětví do domácností a budov. Díky této chytré síti jsme schopni komunikovat s lidmi nebo pracovat se zařízeními na velké vzdálenosti.

Postupem času tak vznikl pojem IoT. Tento pojem označuje síť fyzických zařízení (vozidla, spotřebiče, atd.), která jsou vybavena elektronikou a softwarem se síťovou konektivitou, která umožňuje těmto zařízením propojení a výměnu dat. Každé zařízení je schopno pracovat v infrastruktuře internetu a je v ní také jednoznačně identifikovatelné. Experti odhadují, že do roku 2025 bude do internetu zapojeno až 30 bilionů zařízení. [1]

Značná část chytrých zařízení a technologií se začíná využívat v monitoringu a provozu moderních budov. Měření různých veličin je neoddělitelnou součástí služeb a řízení budov nebo chytrého domu. Někdy se používá pojmenování „smart metering“ (chytré měření), které však není zcela výstižné, jelikož „chytrost“ nespočívá v samotném měření, ale v systému, který data vyhodnocuje a zpracovává [2] V této bakalářské práci se budu zabývat především záznamem, přenosem a prací s daty ze snímačů monitorujících parametry prostředí v budově i mimo ni. Popíši vybraná zařízení pro záznam/zpracovávání dat a navrhnu možné řešení systému monitoringu a řízení budovy s možností využití přístupů umělé inteligence.

1.1 Možnosti pro řízení a monitorování provozu

Budova je nadzemní stavba prostorově soustředěná a navenek převážně uzavřená obvodovými stěnami a střešní konstrukcí. Pro účely analýzy se jedná o objekt, ve kterém byl dokončen aspoň jeden byt. [3]

Řízení chytré budovy a její monitoring je velmi podobný, pro obytné domy, kanceláře a další průmyslové i neprůmyslové budovy. Klíčové je u těchto budov dosažení optimálních hodnot prostředí (např. teploty a vlhkosti) pro jejich uživatele vzhledem k ekonomickému využití energetických zdrojů. [4]

Základem při stavbě budovy je vytvoření centrální ovládací jednotky pro ovládání topení, ventilace, chlazení a světel v budově. Data o jejich provozu pak lze ukládat na paměťová média, kde jsou zdrojem pro rozvoj řídících algoritmů. Tyto tradiční systémy jsou však nezávislé na internetu. Při provozu takových budov se proto vytvoří vnitřní infrastruktura sběrnicového systému (fyzická rozhraní a protokoly pro výměnu dat, např. ISO/OSI) s řídící platformou (např. počítač, mikrokontrolér), které zajistí komunikaci mezi vnitřními systémy.

Komplexnější systémy budov minulých let již integrují snímače/aktuátory od počátečního návrhu projektu na úrovni každého prostoru. Kromě dlouhodobého záznamu dat z vnitřních/venkovních prostor má také velký význam zaznamenávat i další hodnoty např. teplotu užitkové vody. Tyto údaje mohou uživateli poskytnout důležité informace o hospodaření s energií a dlouhodobě snižovat výdaje pro provoz budovy. Současně je důležité uživatele informovat o aktuální spotřebě a dění v domě. Lze podle nich např. vyhodnotit, která okna jsou otevřená a dochází k úniku tepla, nebo zda souběh teploty a vlhkosti nepřesahuje určité procento pro vznik plísně.

Pro příklad možného dlouhodobého využití zaznamenávaných a analyzovaných údajů lze uvést např. rozhodování o možné výměně vnější/vnitřní izolace oken, realizaci dílčí/úplné výměny kotle, instalaci či rozšíření struktury solárního systému nebo rozšíření výkonu tepelného čerpadla.

1.2 Současné možnosti řešení

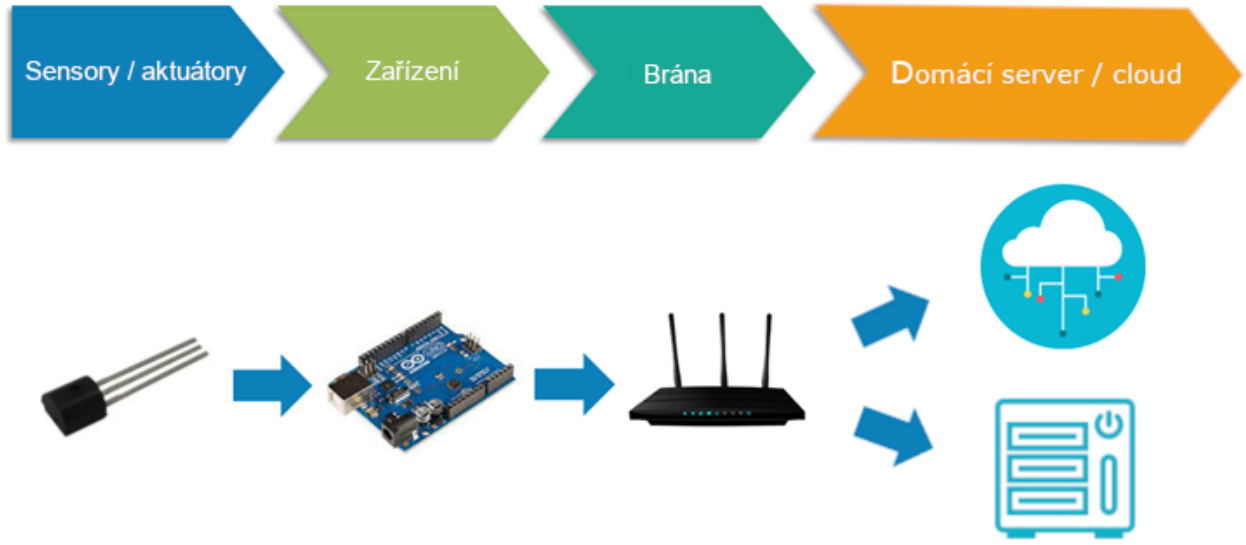
V této chvíli jsou na trhu mnohé systémy pro monitoring a ovládání budovy. Máme možnost si řešení monitoringu navrhnout sami a strávit tak dlouhé dny programováním požadovaného systému nebo si systém pořídit u nějaké firmy. V dalších kapitolách popíši výběr technické platformy, návrh a vývoj řešení včetně popisu možností. V této kapitole dále ve zkratce popíši řešení monitoringu a řízení chytré budovy od společnosti TECO a.s.

TECO a.s. je česká společnost, která vznikla v roce 1993 a zabývá se výrobou průmyslových řídicích systémů kategorie PLC. Tyto systémy je možné nasadit jak do průmyslových, tak do běžných budov. Jejich produkty jsou aplikovatelné v nejrůznějších oblastech automatizace, řízení a ovládání. Další popis společnosti TECO a.s. a jejich chytré budovy lze nalézt v příloze A.

1.3 Potřebné technické prvky pro řízení a monitoring budovy

Klasický model toku dat (nejenom) v budově má zkratku I-P-O. Senzory poskytují vstupy, které se zpracovávají na nějakém počítači nebo mikrokontroléru a spouštějí instalované aktuátory nebo jsou dále posílány do cloudu (nebo jiné datové platformy) a jsou zobrazeny na displejích zobrazovacích zařízení. Tyto informace musí být přenášeny po nějakém datovém médiu, může se jednat např. o vzduch nebo metalické vedení.

Podle technologií přenosu, senzorů nebo aktuátorů může být potřeba přidat k zařízením ještě vstupně/výstupní bránu (en: gateway), často označovanou jako rozbočovač (en: hub). Podobně jako domácí směrovač (en: router) si hub zvládne vyměňovat data s připojenými zařízeními a to drátově, nebo bezdrátově. [4]

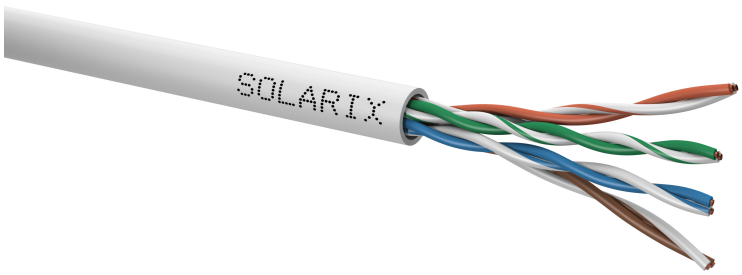


Obrázek 1.1: Základní komponenty pro monitoring a řízení budov, šipky označují přenosové médium

1.3.1 Fyzická vrstva přenosu dat

Základní technický prvek při návrhu pro řízení a monitoring budovy je vytvoření fyzické vrstvy toku dat. Může se jednat např. o strukturovanou kabeláž, nebo bezdrátové připojení. Strukturovaná kabeláž je univerzální kabelový rozvod, který umožňuje přenos digitálních a analogových signálů bez nutnosti další instalace speciálních kabelových rozvodů. Tento systém umožňuje uživateli kdykoliv se rozhodnout, jaká technologie bude použita v konkrétní datové zásuvce (počítač, telefon, IP kamera) a jednoduchým přepojením v datovém rozvaděči změnit směřování konkrétní technologie do daného místa. [5] Normy definují použití 4-párového metalického nebo optického multimode kabelu. V současné době je strukturovaná kabeláž jeden z nejvhodnějších prostředků pro distribuci signálů v rámci budovy. Díky 4-párovému metalickému kabelu můžeme realizovat při extrémním požadavku

na flexibilitu všechny dnes používané slaboproudé aplikace pro přenos dat, hlasu, videa, atd. Tento kabel tak běžně poskytuje distribuci internetového připojení v budově.



Obrázek 1.2: UTP CAT5E kabel společnosti Solarix [6]

Další možnost, jak data v budově přenášet, je pomocí bezdrátové technologie. Jedna z nejpoužívanějších možností je pokrytí prostor budovy technologií Wifi. Wifi je označení pro standardy IEEE 802.11, které definují bezdrátovou komunikaci v počítačových sítích. Wifi sítě nevyžadují pokládku kabelů. Protože signál Wifi má omezenou schopnost šířit se a překonávat pevné překážky, může být spojení méně spolehlivé než v případě kabelového připojení.

1.3.2 Senzory

Senzory jsou elektrické systémy, které nás informují o vlastnostech světa okolo nás. Mohou nás informovat o aktuální teplotě, tlaku, vlhkosti, proudu, poloze, nebo dodat (ne)pravdivou informaci, jestli je např. překročena hladina vody v nádobě. Jejich informace jsou zpracovávány počítačem nebo mikrokontrolérem, který obsahuje ADC a konvertuje tak analogické hodnoty ze senzoru na hodnoty digitální, tedy čitelné počítačem.

Požadavky na senzory

- Závislost výstupní veličiny na veličině vstupní.
- Velká citlivost, přesnost a stálost senzoru.
- Minimální závislost na vlivech okolního prostředí.
- Nízká cena a náklady na provoz.
- Jednoduchá obsluha a údržba.

Obrázek 1.3: Příklad senzoru teploty a vlhkosti [7]

1.3.2.1 Rozdělení senzorů

Senzory se dají rozdělit dle mnoha hledisek. [8]

Základní rozdělení

1. **Senzory elektrických veličin** (proudu, napětí, odporu, ...)
2. **Senzory neelektrických veličin** (polohy, teploty, tlaku, vlhkosti, ...)

Rozdělení dle druhu měřené veličiny

1. **Mechanické** - poloha (délka), úhel natočení, výška hladiny, rychlost, zrychlení, hmotnost, tlak, síla, otáčky, ...
2. **Magnetické** - magnetická indukce, intenzita magnetického pole, magnetický tok, magnetický odpor, ...
3. **Optické** - zářivá energie, intenzita osvětlení, jas, ...
4. **Tepelné** - teplo, teplota, tepelná vodivost, tepelná kapacita, ...
5. **Akustické** - hlučnost, akustický tlak, ...

1.3.2.2 Základní vlastnosti senzorů [9]

1. **Statické vlastnosti senzoru** – vyjadřují vlastnosti při neměnném nebo velmi pomalu měnícím se signálu.
   (a) **Rychlost odezvy** – je určena zejména fyzikálními vlastnostmi senzoru, závisí na rychlosti působení měřené veličiny na převodník.
   (b) **Doba odezvy** – čas potřebný k dosažení určité velikosti signálu v konečném ustáleném stavu

(c) **Šum** – je neužitečný signál, který je vždy modulován na signálu odezvy senzoru. Vzniká zejména průnikem elektromagnetických vln (frekvence 50 Hz rozvodné sítě, měniče napájecích zdrojů).

(d) **Dlouhodobá stabilita**

2. **Dynamické vlastnosti** – vyjadřují chování na rychle měnící se signál.

   (a) **Přechodová charakteristika** – je průběh výstupní veličiny v závislosti na čase při skokové změně vstupní veličiny.

   (b) **Frekvenční charakteristika** - udává závislost přenosu a fázového úhlu na frekvenci, tj. rozdíl amplitudy a fáze výstupního signálu oproti signálu vstupnímu v závislosti na frekvenci.

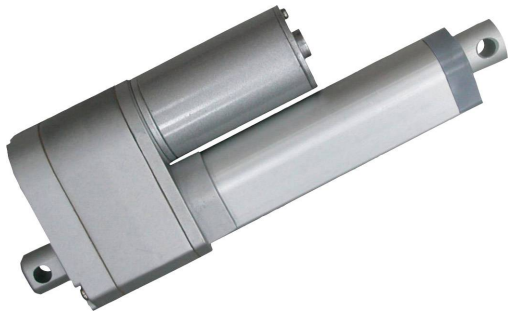
3. **Další vlastnosti** – posun nuly (vlivem teplotního, časového nebo jiného driftu), rušivé vlivy (teplota, tlak, vlhkost, radiace, atd.).

1.3.3 Aktuátory

Aktuátor (akční člen) je obecně zařízení, které poskytuje fyzický výstup na základě informační části procesu primárně formou pohybu. Dobrým příkladem je systém naklápění žaluzií, kdy náklon lamel provádí elektrický pohon, který je nakloní podle informací zaslaných do řídicího systému ze senzorů tepla a světla v místnosti.

Požadavky na aktuátory

- Velká výstupní síla vzhledem k ceně.
- Ideální tuhost aktuátoru.
- Minimální závislost na vlivech okolního prostředí.
- Nízká cena a náklady na provoz.
- Jednoduchá obsluha a údržba.



Obrázek 1.4: Příklad vysouvacího aktuátoru [10]

1.3.3.1 Základní vlastnosti aktuátorů

- **Výstupní síly** – výstupní síla se může lišit pro aktuátory použité v obytných domech a průmyslových budovách, kde jsou nároky na výstupní sílu mnohem vyšší.
- **Bezpečnostní funkce** – aktuátory mohou mít prvek pro navrácení do výchozí pozice v případě chyby.
- **Tuhost aktuátoru** – aktuátor může být ovlivněn změnami tlaku např. regulované kapaliny a jeho tuhost udává sílu, kterou dokáže udržovat např. ventil při tlaku této kapaliny.

1.3.3.2 Rozdělení aktuátorů

Rozdělení aktuátorů podle fyzikálního principu

1. **Elektro-magnetické** – spínání plynu.
2. **Elektromechanické** – využívají elektrické, magnetické, nebo mechanické principy, jejich nevýhodou je závislost na zdroji energie. Mohou také způsobit úraz elektrickým proudem či jiskřením. Jedná se např. o stykač, asynchronní/synchronní motory, stejnosměrné motory, ...
3. **Hydraulické** – výhodou je velká síla, nevýhodou je velikost stroje a nutnost vedení pracovních kapalin. Jedná se např. o přímočaré hydromotory, turbíny, pístová serva.
4. **Pneumatické** – výhodou je rychlost, ekonomická nenáročnost a ekologická nezávadnost. Jedná se např. o ventily, písty, rotační pneumatické motory nebo zpětnovazební soustrojí (klapka-tryska).

Rozdělení aktuátorů podle ovlivněných veličin

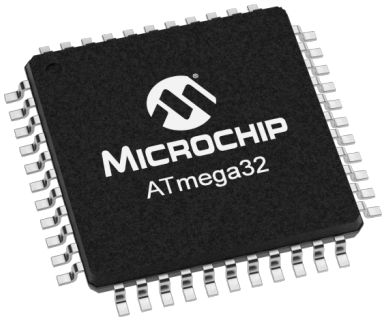
1. **Poloha/rychlost** - v délce, v úhlu.
2. **Síla, moment síly**
3. **Tlak staticky (píst), volný průtok (stavidlo), či obojí najednou** - plynu či kapaliny.
4. **Elektrický proud a napětí** - například při dálkovém přepínání odboček transformátoru.
5. **Zdroje paprsků či osvětlení** - laser, IR.
6. **Průhlednost, zrcadlivost**– např. u LCD.

1.3.4 Zařízení pro zpracování dat

Mikrokontroléry a „systémy na čipu“ (dále také označovány jako vestavěná zařízení) jsou centrem komunikace mnoha senzorů a aktuátorů. Jsou to malé výpočetní zařízení s pamětí a dostupnými vstupními/výstupními porty, které se od běžného počítače liší vykonáváním kódu, jenž je specifický pro úlohy, které jsou na vestavěných zařízeních spouštěny. Zpracování může probíhat také na běžně dostupném stolním počítači, ale jeho provoz by byl v tomto případě sběru informací neekonomický. Pro tento účel se tedy využívají mnohem menší, přemístitelná a jednodušší zařízení, které nepotřebují velký výkon pro zpracování dat ze senzorů.

Mikrokontrolér je jednočipový počítač, který lze různě naprogramovat. Používá CPU, RAM, ROM a vstupy/výstupy procesoru. Vše je v tomto případě integrované v jednom čipu.

Vestavěná zařízení jsou využita v elektronických zařízeních, které potřebují elektrické ovládání. Jejich použití lze nalézt všude okolo nás od zařízení v domácnostech (lednička, pračka, mixér, zvonek), až po využití v průmyslových strojích. Dnešní auta v sobě také mají desítky až stovky propojených vestavěných zařízení, které např. zaznamenávají různé hodnoty ze senzorů a předávají je hlavní výpočetní jednotce automobilu.



Obrázek 1.5: Mikročip AVR ATmega32 [11]

1.3.5 Brány a rozbočovače

Brány, často označované jako rozbočovače, slouží jako prostředník mezi senzory a platformou pro zpracování a uložení dat. Jejich funkce se v různých systémech liší, ale obecně slouží k předzpracování, formátování a následného poslání dat do vyšší vrstvy systému. Jsou velmi efektivní v systémech, které obsahují četnou řadu senzorů. Brána/rozbočovač totiž k poslání dat ze všech senzorů potřebuje jen jedno připojení, čímž se šetří zdroje (energie, rychlost přenosu) a nároky na přijímací platformu. Kdyby se každý senzor (díky např. mikrokontroléru) připojoval k přijímací platformě zvlášť, musela by zvládat mnohem více požadavků a plýtvalo by se tak zdroji. Brány/rozbočovače tedy v některých systémech nutné nejsou, ale jejich vlastnosti dokáží v některých případech zefektivnit práci s daty.

Obrázek 1.6: Příklad brány, směrovač Cisco RV160W WLAN AC [12]

1.3.6 Cloudy a datové platformy

Cloud je v současné době velmi populární pojem. Je to synonymum pro počítačový zdroj, který je přístupný skrze internet. Na internetu existuje velké množství poskytovatelů, kteří mají k dispozici datová centra obsahující stovky fyzických počítačů, jejichž zdroje, úložiště a další funkce si může člověk zakoupit. Výhodou je, že cloud je v tomto případě virtuální a člověk se tak nemusí starat o zařízení, na kterých cloud běží. Fyzické místo (např. datové centrum), kde jsou cloudová data fyzicky uložena může být kdekoliv na světě. Mezi největší poskytovatele cloudových služeb patří Google (se službou Google drive), Microsoft (se službou Microsoft OneDrive) a Dropbox. Jejich ceny se liší podle velikosti úložiště a poskytovaných služeb. K roku 2021 se ceny za 1TB dat pohybují okolo 200 Kč měsíčně.

Mezi základní funkcionalitu patří úložný prostor pro data. Toho lze využít pro ukládání dat zaznamenaných na senzorech. Dále také cloudy poskytují různé funkce jako vizualizace dat, nebo poskytují výpočetní výkon např. pro ovládání částí chytré budovy. Cloud ale nemusí běžet jen na počítačích v datových centrech. Lze ho také vytvořit doma např. ze starého počítače, serveru, nebo si pro tento účel koupit síťové úložiště (NAS). Síťové úložiště je druh počítače, který se specializuje na ukládání a práci s daty po síti. Poskytuje většinou stejnou funkcionalitu jako virtuální cloudy. Pořizovací cena síťového úložiště se velmi liší podle velikosti paměti a funkcionality. Jako základní příklad síťového úložiště lze uvést např. Synology DS220+, který se průměrně k roku 2020 prodává za 10 000 Kč s jedním 1TB diskem. V mém systému byly vyzkoušeny oba přístupy a jejich dalším srovnáním (bezpečnost, cena, rychlost, atp.) se budu více zabývat v následujících kapitolách.

1.4 Volba budovy pro experimentální ověřování

Jako budovu pro monitoring jsem vybral vlastní dům z důvodu dobré dostupnosti a znalosti vnitřních systémů. Jedná se o třípatrový rodinný dům, který je vytápěn tepelným čerpadlem ve spojení s podlahovým topením. V místnostech s keramickým povrchem jsou navíc instalovány elektrické

topné rohože. Tato budova před začátkem vytváření mého systému neobsahovala chytré systémy pro monitoring nebo automatizaci.

Obrázek 1.7: Fotografie budovy pro experimentální ověřování

V domě je rozvedená strukturovaná kabeláž, která zajišťuje přenos dat do většiny místností v budově. Dům je bezdrátově připojen do veřejné části internetu přes RouterBoard spolku KlimNet z.s. RouterBoard (lze jej vidět na obrázku 1.8) je označení pro malé základní desky od firmy MikroTik, které lze využít jako přístupové body, směrovače nebo mosty. V tomto případě je toto zařízení připojeno jako most a příchozí provoz přesměrovává na domácí směrovač společnosti Cisco RW110. Tento směrovač je dále napojen na síťový přepínač (en: switch), do kterého jsou v domě připojeny všechny Ethernetové LAN zásuvky a dva přístupové body poskytující kvalitní pokrytí technologií Wifi ve většině prostor uvnitř i v okolí domu. Topologie původní a navržené domácí sítě je zobrazena a více popsána v kapitole 3.

Obrázek 1.8: RouterBoard poskytovatele pro připojení budovy do veřejné části internetu

Elektřina je do budovy přivedena podzemním vedením z podružného rozvaděče, kde jsou umístěny elektroměry. Tento rozvaděč je umístěn v zídce před domem a rozvádí elektřinu do dalších domů v ulici. Podružný a hlavní rozvaděč lze vidět na obrázku 1.9.

(a) podružný rozvaděč (b) hlavní rozvaděč v budově

Obrázek 1.9: Podružný a hlavní rozvaděč

Kapitola 2

Výběr technické platformy, návrh a vývoj řešení

Základem pro úspěšné měření teploty, vlhkosti a tlaku je výběr senzorů a zařízení, která data ze senzorů budou zpracovávat. V současné době je na trhu mnoho zařízení, kterými se z okolního světa dají zpracovávat a číst data. Díky dostupnosti se snižuje cena zařízení a zmenšila se i jejich velikost. Díky tomu dnes lze s mikročipem o velikosti desítek milimetrů ovládat několik zařízení najednou a zpracovávat jím různá data.

2.1 Zařízení pro zpracovávání dat a řízení

V současné době jsou mezi vestavěnými zařízeními na vzestupu mikrokontroléry Arduino a Raspberry Pi. Dají se jednoduše naprogramovat přes USB rozhraní a jejich rozměry jsou velmi malé (velikost kreditní karty). Jejich úspěch vedl k velkému rozšíření těchto platforem i mezi běžnou populaci a výrobci tak začali k zařízením vytvářet i různé nadstavby, které základnímu zařízení poskytují rozšířenou funkčnost jako GPS, Ethernet připojení nebo rozhraní pro různé systémové sběrnice. V profesionálním měřítku jsou ale jen vzácně nasazeny kvůli jejich velikosti a náchylnosti na různé pracovní podmínky, jako prach, teplota, nebo vlhkost. Jsou ale velmi používané jako zařízení pro výrobu prototypů a testování funkčnosti. [4]

Pro záznam dat a vytváření IoT zařízení se kromě čipů Arduina a Raspberry Pi často využívají i mikročipy, které v základu poskytují bezdrátovou možnost přenosu dat. Jedná se např. o mikročipy firmy Espressif Systems – ESP8266 a ESP32.

Arduino UNO

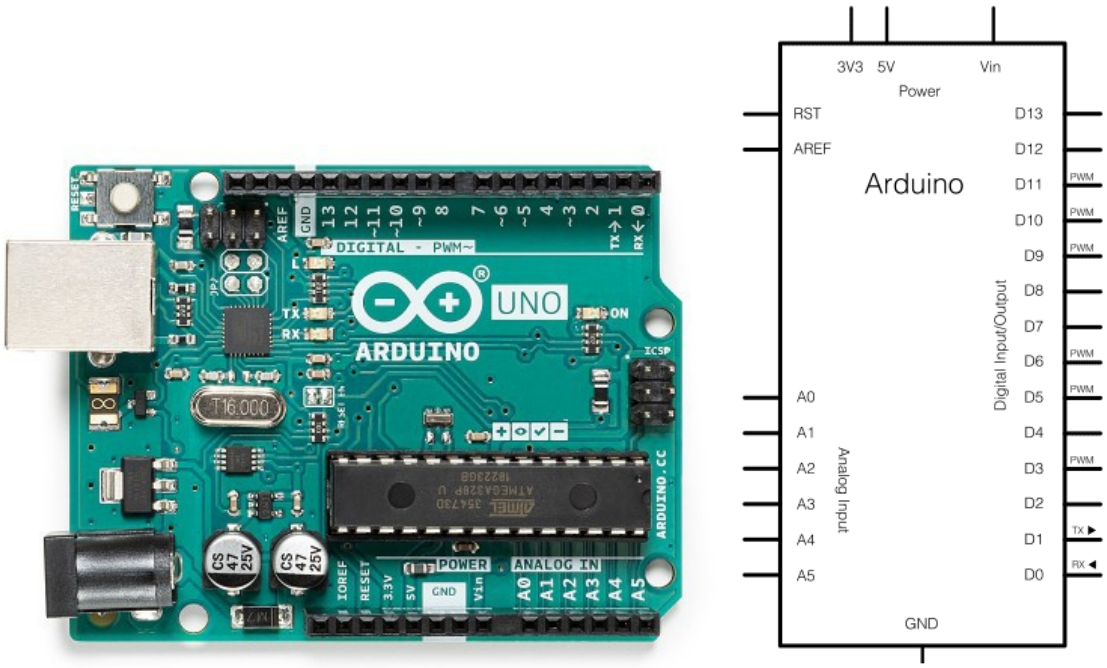
Arduino bylo vyvinuto v Institutu Ivrea Interaction Design v roce 2005 jako zařízení pro rychlé prototypování mířené zejména pro studenty a příznivce elektroniky bez znalosti programování. Brzy

tato uživatelsky přívětivá platforma přitáhla pozornost široké veřejnosti a zcela tak změnila trh elektronických zařízení. Zdrojový kód Arduina je také kompletně otevřený a umožňuje tak všem jeho uživatelům měnit a dále distribuovat zdrojový kód.

Tento mikrokontrolér nemá operační systém a provádí cyklicky jeden program, který je do něj nahrán. Jeho hlavní výhodou je napájení přes USB (přes USB je také zajištěn přenos dat mezi počítačem) i klasickými baterkami.

Klíčové vlastnosti

1. Nízká cena, základní deska Arduino UNO stojí přibližně 200 Kč a lze ji nalézt v každém obchodě s elektronikou.
2. Jeho software je přívětivý uživatelům a jednoduchý pro rychlý start vývoje elektroniky. Může být spuštěn na operačních systémech Mac, Linux i Windows.
3. Podporuje několik programovacích jazyků, mezi hlavní se řadí C a C++.



Obrázek 2.1: Arduino UNO [13] a jeho blokový diagram

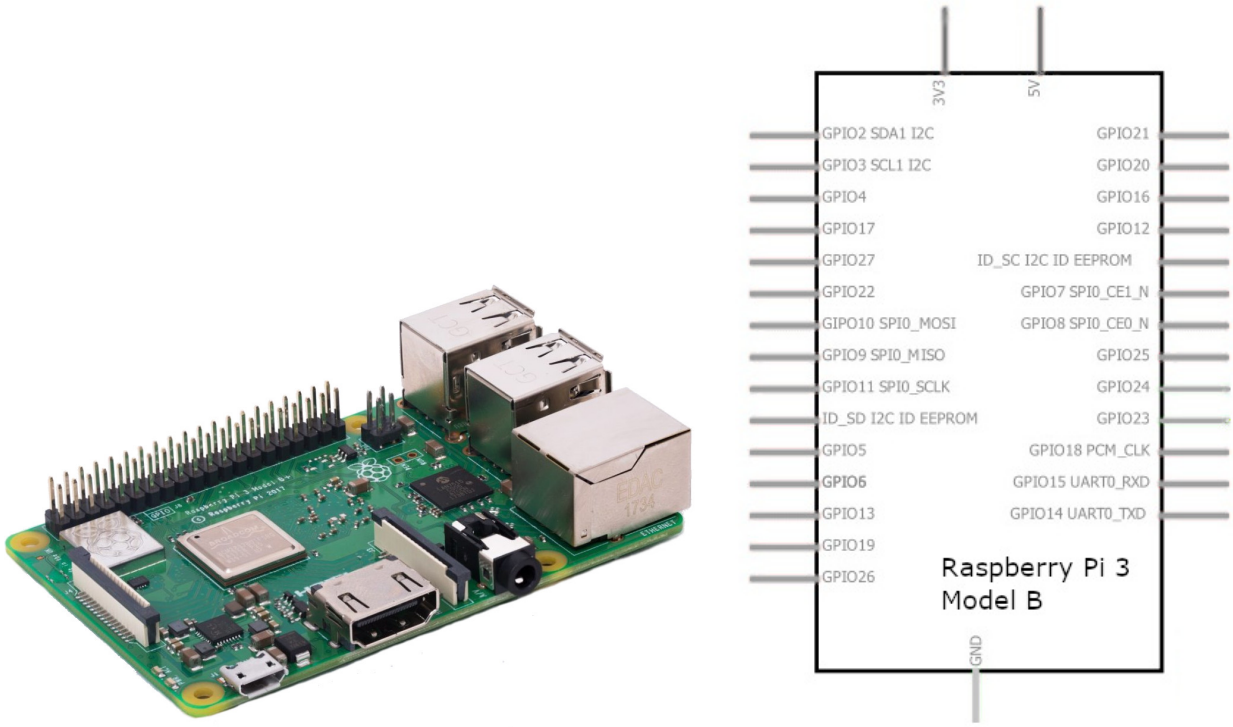
Raspberry Pi 3

Raspberry Pi byl vyvinut britskou nadací Raspberry Pi Foundation s cílem podpořit výuku informatiky ve školách v roce 2012. Oproti Arduinu se řadí do kategorie malých levných počítačů a vyznačuje se tím, že obsahuje operační systém Raspbian OS, který zvládá vykonávat více úloh

najednou a udržovat v běhu více programů zároveň. Lze jej tedy připojit k televizi nebo monitoru a ovládat pomocí periferií jako myš či klávesnice. Vzhledem ke složitější architektuře je vyšší i pořizovací cena (okolo 1200 Kč s 1Gb pamětí).

Klíčové vlastnosti

1. Lze jej použít na komplexnější problémy, kde je třeba větší výpočetní výkon a potřeba běhu více aplikací zároveň.
2. Lze jej jednoduše připojit k internetu bez nutnosti přidání pomocného hardware.
3. Poskytuje port na SD kartu pro ukládání dat.
4. Lze na něj programovat ve více jazycích jako Python, C, C++, které jsou již předinstalované.



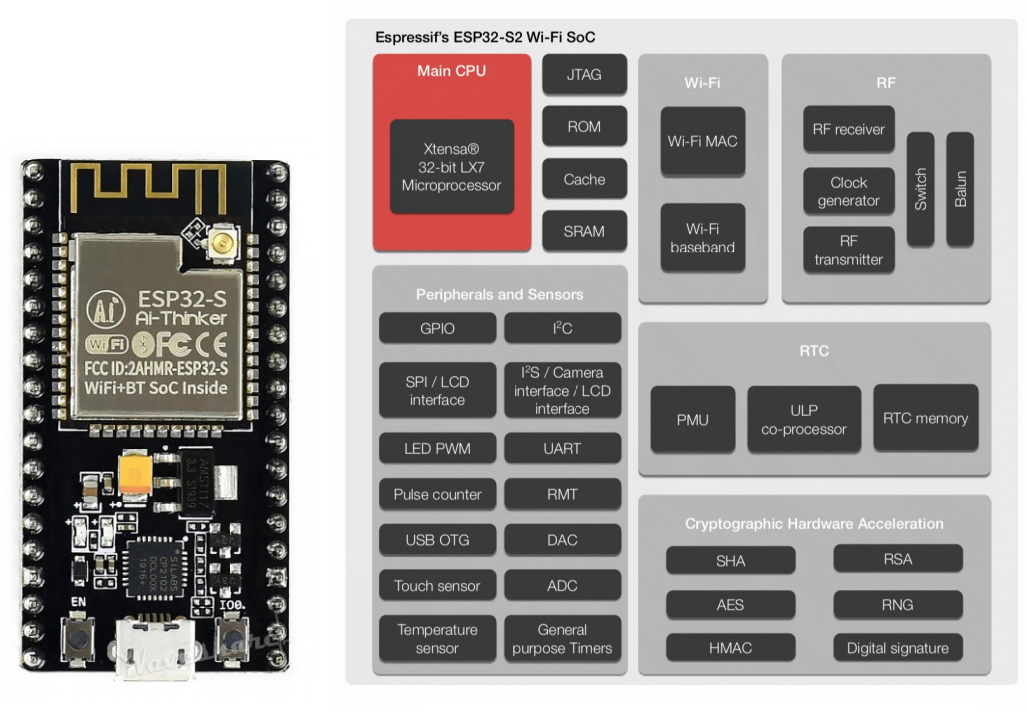
Obrázek 2.2: Raspberry PI model 3 [14] a jeho blokový diagram

NodeMCU-S ESP32

NodeMCU-S s čipem ESP32 je moderní mikrokontrolér od firmy Espressif Systems a nástupce ESP-01 (s čipem ESP8266). Jeho hlavní výhoda jsou zabudované technologie Wifi a Bluetooth. Není třeba tedy dokupovat nadstavbu pro připojení do internetu, jako v případě Arduina. Tento mikrokontrolér je za dostupnou cenu a má v sobě zabudované velké množství funkcí. Je často využíván pro IoT projekty v průmyslu i domácnostech.

Klíčové vlastnosti

1. Lze jej programovat přes Arduino IDE v jazyce C++.
2. Má v sobě zabudovaný standart 802.11b/g/n Wifi, Bluetooth 4.2 a Bluetooth Low Energy.
3. Obsahuje vnitřní obvod RTC k časování a uspávání/probouzení zařízení.
4. Podporuje spánek zařízení k šetření spotřeby proudu.
5. Nahrávání programů a komunikace probíhá přes USB-B.



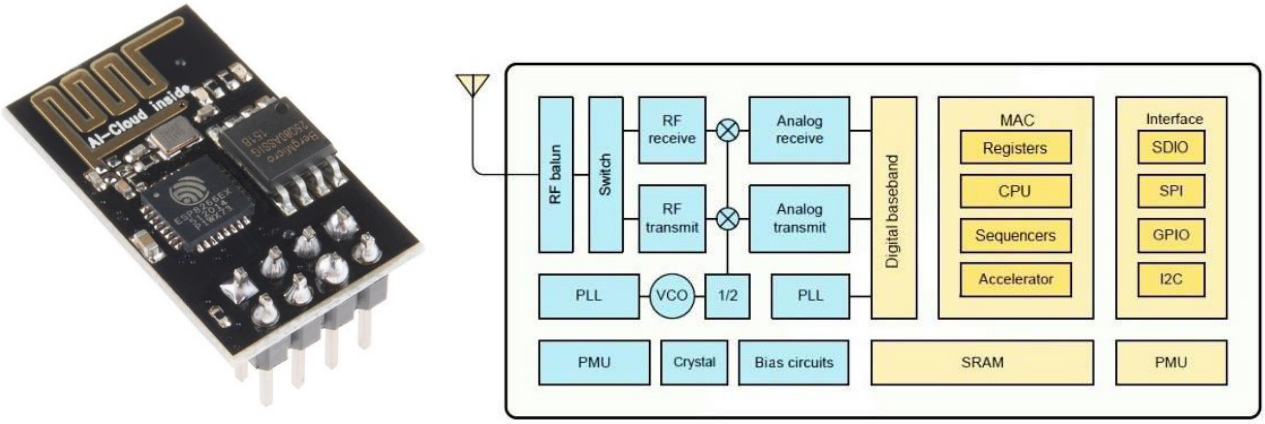
Obrázek 2.3: Mikrokontrolér NodeMCU-S ESP32S [15] a blokový diagram jeho čipu ESP32 [16]

ESP-01

ESP-01 je mikrokontrolér s Wifi mikročipem ESP8266 od firmy Espressif. Je velmi levný a nabízí kompletní a ucelené Wifi síťové řešení. Lze jej naprogramovat přes UART sběrnici pomocí Arduina nebo samostatného UART modulu. Po naprogramování umí fungovat dále samostatně. Pracuje na internetovém protokolu TCP/IP a lze jej tak použít pro projekty v oblasti IoT.

Klíčové vlastnosti

1. Nízká cena, k roku 2021 okolo 80 Kč.
2. Malé kompaktní rozměry, má ale jen dva piny pro ovládání zařízení.
3. Je složitější na zapojení, nedá se jen tak jednoduše připojit do nepájivého pole.



Obrázek 2.4: Mikrokontrolér ESP-01 [17] a blokový diagram jeho čipu ESP8266 [18]

2.2 Senzory pro měření a záznam dat

Senzor je obecně zařízení, které konvertuje data z reálného světa do dat požadovaného systému pomocí ADC převodníků. Díky senzorům můžeme vyčíst velké množství informací z okolí a následně je zpracovat dalšími zařízeními. V této kapitole se budu zabývat senzory, které byly vyzkoušeny nebo použity v mém systému.

Kritéria pro výběr senzoru

Před koupí senzoru je třeba zvážit několik věcí. První důležité kritérium je cena, která se může pohybovat od desítek, po stovky korun. Většinou se od ceny odvíjí i přesnost senzoru. Je důležité dopředu vědět, jestli např. odchylka teploměru o dva stupně je pro systém přípustná. Jako další kritérium jsou provozní podmínky senzoru. Teplotní čidlo s rozsahem 0 °C až 65 °C není dobré umisťovat ven, kde může být -20 °C a vystavovat tak senzor riziku zničení. Je důležité tedy zvážit všechna kritéria a promyslet, v jakém systému bude senzor nasazen.

2.2.1 Senzory pro měření teploty

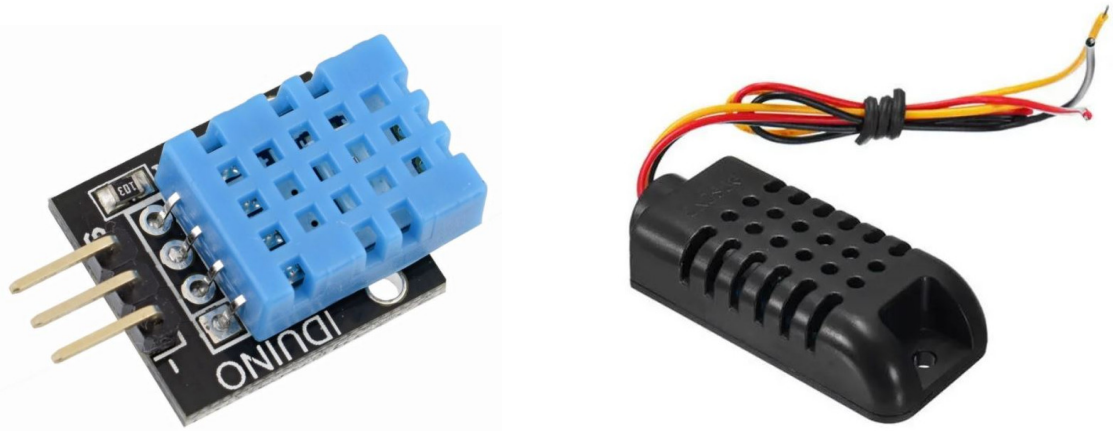
Teplota je fyzikální veličina vyjadřující míru tepelného stavu tělesa. K měření se nejčastěji využívá teplotní stupnice termodynamická (Kelvinova, jednotkou je Kelvin [K]) a mezinárodní ITS-90 (Celsiova, jednotka je stupeň Celsia [°C]). Senzory pro měření teploty můžeme rozdělit na **senzory pro dotykové měření** (elektrické, dilatační, tlakové) nebo **senzory pro bezdotykové měření** (monochromatické, pásové a radiační pyrometry). Elektrické senzory teploty se dále dělí na termoelektrické, polovodičové, odporové kovové a odporové polovodičové. U každého senzoru – zjistit, jak jsou vyrobeny.

DHT11

Při vybírání prvního zařízení pro záznam dat teploty a vlhkosti do mého systému byl poprvé použit senzor DHT11. Je levný (okolo 40 Kč) a kromě teploty umí měřit i vlhkost. Je také jednoduchý k použití, ale požaduje, aby záznam dat měl správné časování. Nová data lze ze senzoru číst dvě sekundy po předchozím měření. Tato prodleva může být v některých projektech moc dlouhá. Pro měření teploty využívá termistor a řadí se tak mezi polovodičové snímače teploty. Pro měření vlhkosti využívá kapacitní princip. Může pracovat v teplotách od 0 °C až 60 °C. Výrobce udává odchylku ± 2 °C, ale při reálném měření měl senzor odchylku větší. Jeho udávaný rozsah měření vlhkosti je 20 % až 90 % s odchylkou ± 5 %. Při reálných měřeních odchylka dosahovala až 10 %. Kvůli nedostatečné přesnosti a špatné funkcionalitě byla do mého systému proto zvolena jeho vyšší verze DHT21, která svými parametry plně vyhovovala požadavkům.

DHT21

DHT21 je vyšší verze DHT11, která poskytuje větší rozsah teplot měření (-40 ℃ až 80 ℃) a vyšší přesnost měření (okolo ± 0,5 ℃). Kromě měření teploty umožňuje také měření vlhkosti. Pro měření teploty a vlhkosti využívá stejné principy jako DHT11. Jeho cena je vyšší než u DHT11 (okolo 130 Kč), ale při měření teplota i vlhkost odpovídaly reálným hodnotám v místnosti. V mém systému jsem tento senzor využil při měření teploty a vlhkosti ve dvou patrech budovy.



Obrázek 2.5: Senzory DHT11 [19] (vlevo) a DHT21 [7] (vpravo)

DS18B20

DS18B20 je kvalitní teplotní senzor, který měří s přesností až 0,2 °C. Teplotní rozsah má od -55 °C do 125 °C. Vydrží tedy venkovní podmínky i v zimě v našich klimatických podmínkách. Pro měření teploty využívá termistor a řadí se tak mezi polovodičové snímače teploty. Senzor je také poměrně levný s cenou okolo 50 Kč. Vyrábí se i ve voděodolné verzi, která byla v mém systému využita

pro měření teploty tepelného čerpadla. Jeho odchylka byla v řádu desetin. Jeho velkou výhodou je možnost zapojení více těchto senzorů na jednu sběrnici (OneWire), čehož bylo v mém systému také využito.

Obrázek 2.6: Senzory DS18B20+ [20] (vlevo) a DS18B20 [21] (vpravo)

2.2.2 Senzory pro měření vlhkosti

Vlhkost vzduchu je veličina, která udává, jakou měrou je vzduch nasycen vodními párami. Je vyjadřována v procentech a rozlišujeme ji na absolutní, relativní a měrnou. Pro vyhodnocování vlhkosti se používají různé metody (psychometrická, hygrometrická, kapacitní, rosného bodu, coulometrická, atd.).

DHT11

Tento senzor byl popsán v sekci Senzory pro měření teploty výše v dokumentu.

BME280

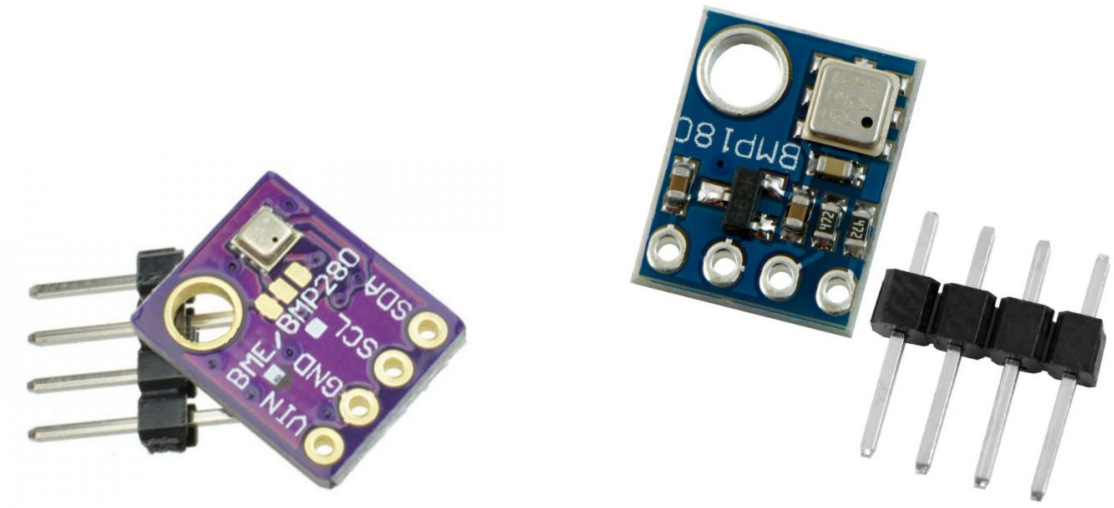
Tento senzor bude popsán v sekci Senzory pro měření tlaku níže v dokumentu.

2.2.3 Senzory pro měření tlaku

Tlak vzduchu (atmosférický tlak) je síla, která působí v daném místě na plochu jednotkové velikosti. Je vyvolán tíhou vzduchového sloupce, který sahá od hladiny moře (nebo jiné sledované výškové hladiny) až k horní hranici atmosféry. Tlak vzduchu se měří v pascalech [Pa] a k jeho měření se využívají tlakoměry. Tlakoměry se dělí na deformační, kapacitní a tzv. tenzometry. Pro měření atmosférického tlaku se často využívají deformační tlakoměry s membránou.

BME280

Senzor BME280 je kombinací teploměru, vlhkoměru, tlakoměru a využívá I2C sběrnici. Teplotní rozsah měření má od -40 °C do 85 °C s přesností okolo ±1 °C. Jeho rozsah měření vlhkosti je 0 % až 100 % s odchylkou ± 3 %. Tlak měří v rozsahu od 300 hPa do 1100 hPa s přesností ±1 Pa. Senzor funguje na principu deformace vnitřní flexibilní membrány a řadí se tak mezi deformační tlakoměry. Cena se pohybuje okolo 170 Kč. Tento senzor jsem využil pro měření teploty, tlaku a vlhkosti v okolí domu kvůli jeho malým rozměrům a dobré přesnosti měření všech měřených veličin. Kromě tohoto senzoru byl v mém systému vyzkoušen i jeho předchůdce, senzor BMP180, který je velmi podobný, ale dokáže měřit jen teplotu a tlak. Senzor BMP180 má nižší přesnost, ale také nižší cenu (okolo 100 Kč). I přes nižší cenu však senzor BMP180 nebyl použit v mém systému z důvodu velké odchylky při měření tlaku (byla u něj naměřena odchylka zhruba 100 hPa).



Obrázek 2.7: Senzory BME280 [22] (vlevo) a BMP180 [23] (vpravo)

Kapitola 3

Návrh řešení pro měření teplot, tlaku a vlhkosti v budově a exteriéru

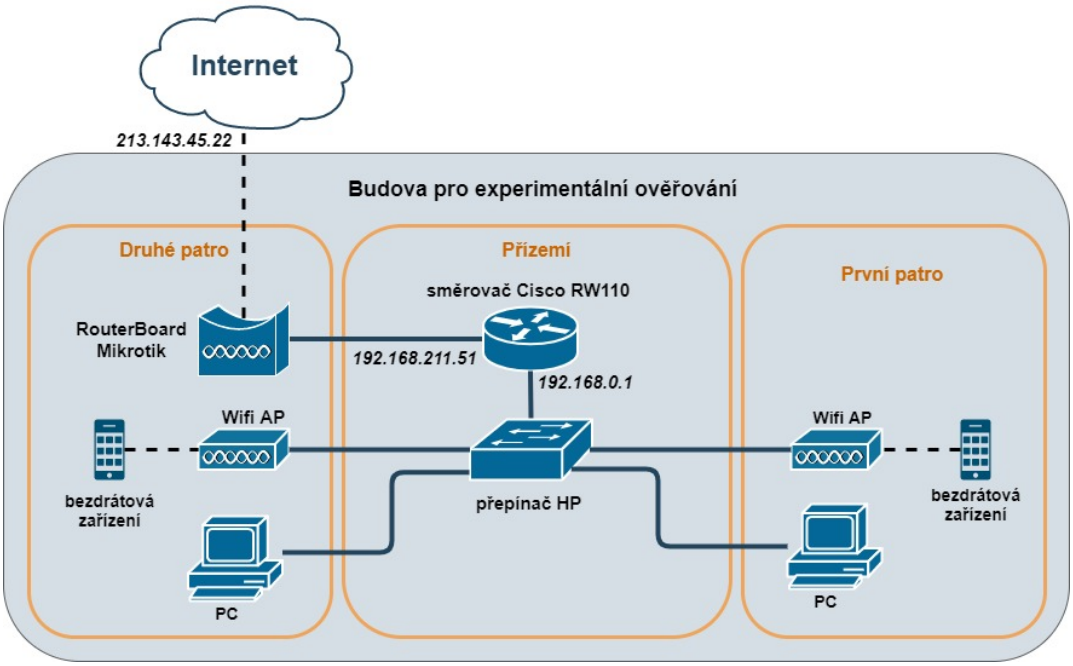
3.1 Popis původního systému

Původní systém v experimentální budově neměl před začátkem vytváření mého systému žádné podsystémy pro monitoring nebo záznam dat. Jeho výhodou ale bylo dobré pokrytí drátového a bezdrátového datového připojení ve všech patrech budovy i v okolí domu. Této vlastnosti bylo v mém systému využito a v každém patře tak byl vytvořen malý systém pro monitoring. Původní topologii sítě lze vidět na obrázku 3.1

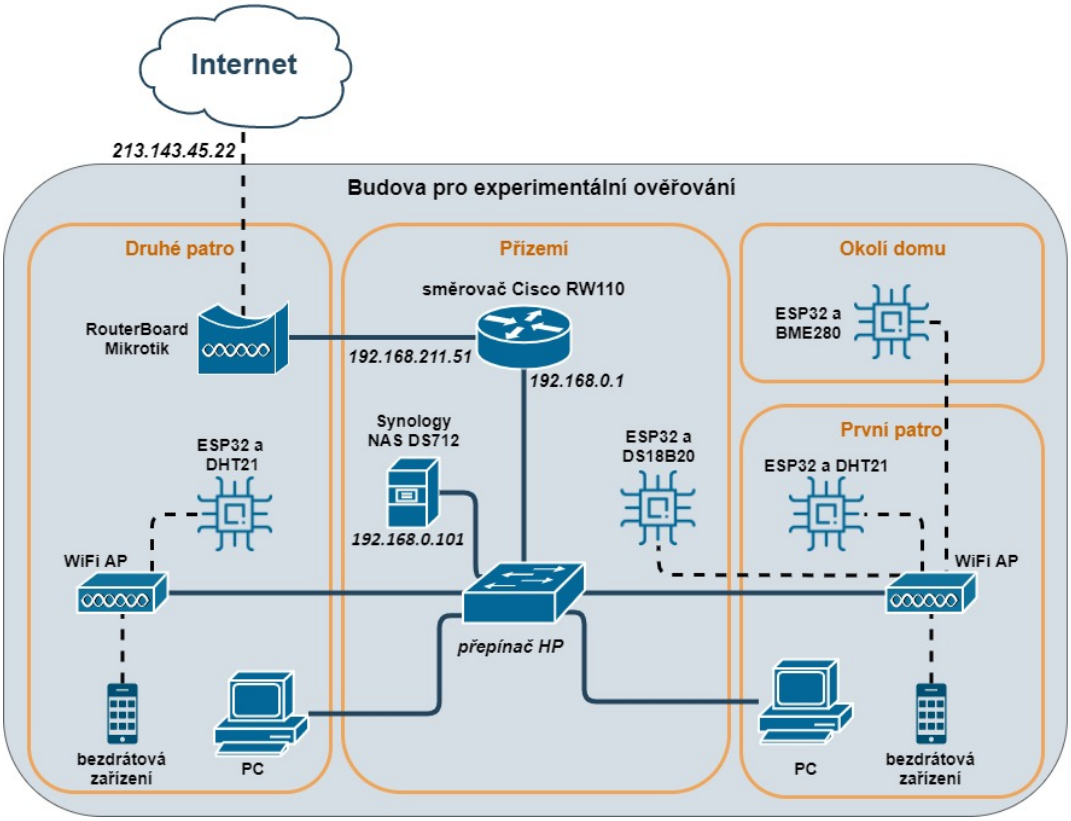
3.2 Popis navrženého systému

Systém pro řízení a monitoring byl rozdělen na 3 části (podsystémy). První dva podsystémy byly umístěny dovnitř budovy a třetí do venkovního prostoru budovy. Topologie navržené sítě je zobrazena na obrázku 3.2. Na tomto obrázku lze vidět, že kromě jednotlivých částí systémů pro monitoring přibyl také oproti původnímu systému lokální NAS server. Na tomto serveru se budou v mém systému ukládat všechna data ze senzorů a více bude popsán ve třetí kapitole.

Po rozvržení jednotlivých částí systému musel být vybrán mikrokontrolér, který by data uměl v pravidelných intervalech číst a posílat přes domácí síť na lokální NAS server. Bylo třeba zvážit, jestli budou data posílána přes pevnou linku nebo bezdrátově. V systému jsem zvolil posílání dat bezdrátově kvůli dobré dostupnosti bezdrátové sítě ve většině prostor budovy. Při výběru mikrokontroléru přišlo v úvahu Arduino, jenže samotné Arduino pracovat se sítí neumí. K Arduino by musela být zakoupena nadstavba. Z důvodu vysoké ceny do systému nebyl nasazen ani mikrokontrolér Raspberry PI. Do mého systému byl nakonec vybrán mikrokontrolér NodeMCU-S ESP32 kvůli jeho kompaktním rozměrům, dostupné ceně a velkému množství funkcí (RTC, USB, možnost spánku zařízení), které jsem mohl v systému využít.



Obrázek 3.1: Původní topologie sítě a k ní připojená zařízení



Obrázek 3.2: Navržená topologie sítě a k ní připojená zařízení

3.2.1 Podsystém pro monitoring teploty a vlhkosti v budově

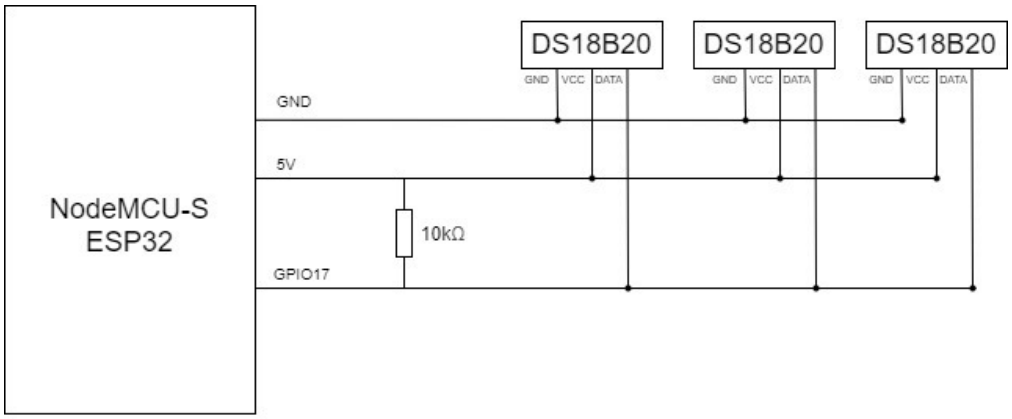
První část byla vytvořena pro monitoring teploty a vlhkosti v prvním a druhém patře budovy. K tomuto účelu byl využit senzor DHT21 kvůli jeho dobré přesnosti měření a jednoduchého zapojení. Tlak v budově v této části systému nebyl měřen z důvodu následné nevyužitelnosti tlakových hodnot při návrhu vyhodnocování závěrů umělé inteligence v domě. Na obrázku 3.3 je zobrazeno schéma zapojení této části systému.

Obrázek 3.3: Systém pro záznam dat vlhkosti a teploty uvnitř budovy a jeho blokové schéma zapojení

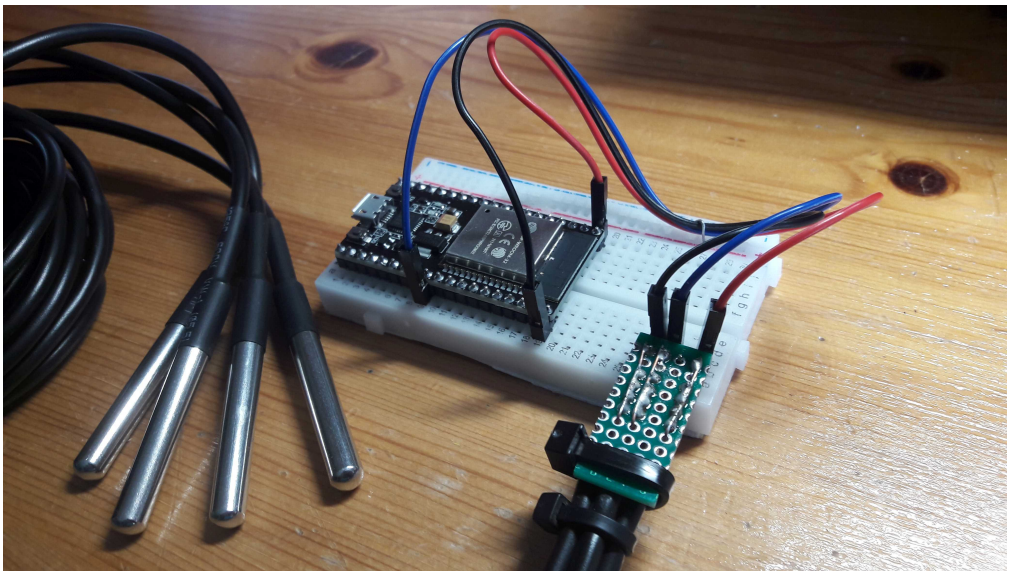
3.2.2 Podsystém pro monitoring tepelného čerpadla

Druhá část byla zaměřena na snímání teploty tepelného čerpadla v domě. Jako senzor byl použit DS18B20 ve voděodolné verzi z důvodu jednoduché instalace jeho sondy přímo na tepelné čerpadlo. Dále byl zvolen pro jeho velkou přesnost při měření a dostupné ceně. Tepelné čerpadlo v domě je typu země-voda a obsahuje vnitřní a vnější okruh. Vnitřní okruh přivádí teplou vodu do podlažního topení po celé budově. Vnější okruh vodu žene do hloubkového vrtu, kde se voda mírně ohřeje a vrací se zpátky do domu. Výměna tepla probíhá přes tepelný výměník. Senzory jsou přiloženy k oběma trubkám vnějšího i vnitřního okruhu. Díky těmto čtyřem senzorům lze monitorovat, jaká teplota vody jde do/z domu. Při vyhodnocování algoritmů umělé inteligence by těchto hodnot šlo využít ke snížení provozních nákladů tepelného čerpadla, protože by bylo možné optimalizovat jeho provoz a výkon.

Blokové schéma zapojení této části systému je zobrazeno na obrázku 3.4. Velkou výhodou u senzoru DS18B20 je možnost zapojení více senzorů na jedné datové lince. Tato funkcionalita je možná, protože každý senzor má v sobě od výroby zapsanou jedinečnou adresu, podle které pak sběrnice může přijímat data. Do obvodu je také připojen 10 kΩ „pull-up“ rezistor, aby byl přenos dat stabilní. Obrázek reálného zapojení k tepelnému čerpadlu je zobrazen v příloze D.



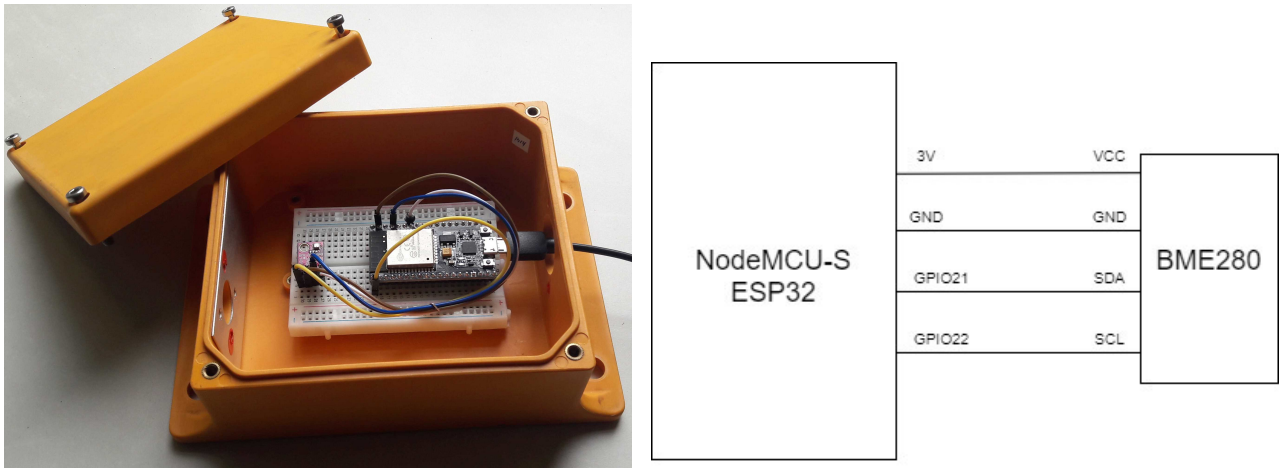
Obrázek 3.4: Blokové schéma zapojení systému pro monitoring tepelného čerpadla



Obrázek 3.5: Systém pro monitoring tepelného čerpadla

3.2.3 Podsystém pro monitoring teploty, tlaku a vlhkosti v okolí budovy

Třetí část byla vytvořena pro snímání venkovní teploty, tlaku a vlhkosti. K tomuto účelu byl použit senzor BME280 z důvodu dobré přesnosti měření všech veličin, malých rozměrů a dobrého pracovního rozsahu teplot. Teplotní rozsah je u venkovních systémů důležitým kritériem z důvodu vystavení senzoru jeho hraničním teplotám. V zimě u nás mohou dosahovat teploty až -20 °C. Tento senzor má ale maximální zápornou pracovní teplotu -40 °C a pro podnebí v České republice v nížinách tedy dostačuje. Na obrázku 3.6 je zobrazeno schéma zapojení této části systému, na obrázku lze také vidět, že systém je připraven ve voděodolné krabičce pro venkovní použití a možnost instalace na zahradní zídku.



Obrázek 3.6: Systém pro záznam dat vlhkosti a teploty uvnitř budovy a jeho blokové schéma zapojení

3.3 Program pro sběr teploty a vlhkosti

Mikrokontrolér NodeMCU-S lze připojit k počítači přes USB-B rozhraní, přes které se dá programovat i ladit. Po zapojení hardware části systému musel být vytvořen program, který v pravidelných intervalech bude číst data ze senzorů a posílat tato data na webový server, který je dokáže přijmout, uložit a odeslat dalším zařízením. Program byl napsán v jazyce C++ ve vývojovém prostředí Arduino IDE, které podporuje nahrávání programů do mikrokontroléru NodeMCU-S ESP32.

Program dále popsaný byl použit pro všechny tři části systému s mírnými rozdíly v programovém kódu. Jako následný příklad jsem využil podsystém pro záznam teploty a vlhkosti v prvním patře. Programový kód pro další senzory lze nalézt v přílohách B a C.

K odesílání dat z mikrokontroléru bylo využito technologie Wifi, kterou mikrokontrolér disponuje. O záznam teploty a vlhkosti se stará senzor DHT21, který čte teplotu a vlhkost v místnosti. U každého záznamu musí být také čas a datum měření. Zařízení tedy získává i informaci o aktuálním čase z NTP serveru, ke kterému se ESP32 před začátkem měření připojí. Měření probíhají v patnáctiminutových intervalech.

Inicializace globálních proměnných a import knihoven

Na začátek programu je nutné importování knihoven pro práci s vestavěnými moduly zařízení, nastavení pinu pro příjem dat z DH21 a zadání platných verifikačních údajů pro připojení k Wifi.

```
#include <WiFi.h>    // knihovna pro komunikaci mezi serverem a ESP32
#include <ArduinoJson.h> // knihovna pro práci s formátem JSON
#include "DHT.h"     // knihovna pro čtení hodnot z DHT21 senzoru
#include "time.h"    // knihovna pro práci s časem
const char* ssid   = "jmeno_wifi";
const char* password = "heslo_wifi";
```

Listing 3.1: Soubor: ESP32_deep_sleep.ino

Dále bylo třeba nastavit adresu NTP serveru, ze kterého si zařízení čte údaj o čase. Jelikož se Česká republika nachází v časové zóně GMT+1, musel být nastaven „offset“ o 3600 sekund (1 hodina).

```
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 3600; // posun času
const int  daylightOffset_sec = 3600;
```

Listing 3.2: Soubor: ESP32_deep_sleep.ino

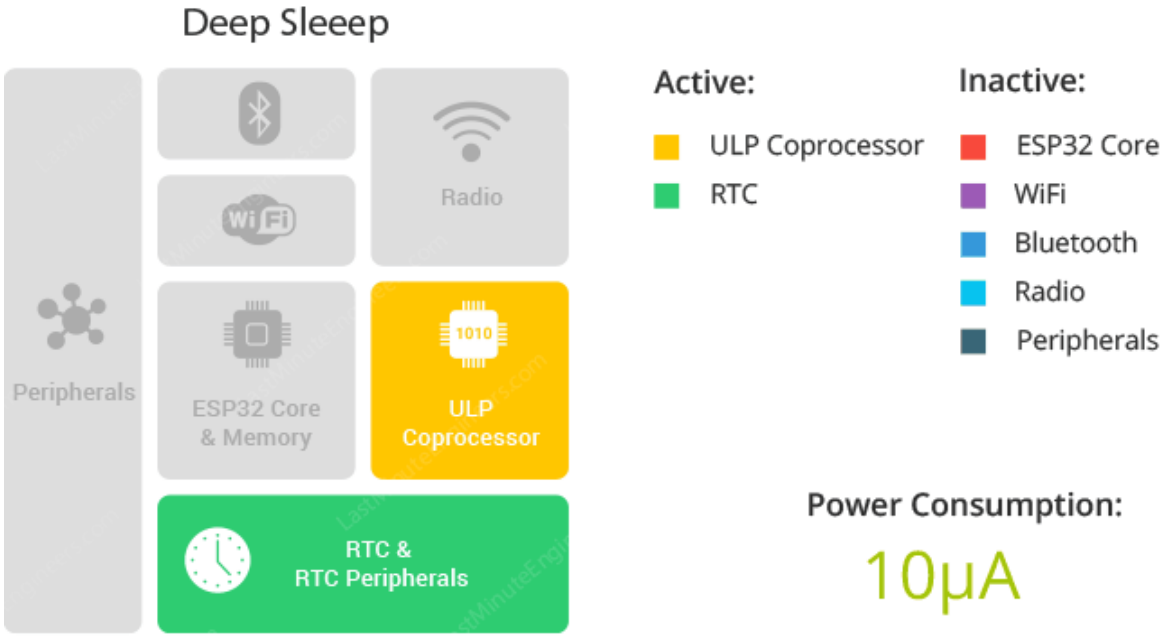
Pro jednoduchou přemístitelnost je možné zařízení napájet z přenosného zdroje elektrické energie (en: powerbank) nebo bateriemi. Do systému byl vybrán přenosný zdroj o kapacitě 3000 mAh. Technologie Wifi je ale velmi náročná co se týče spotřebovávání elektrické energie, a proto byla využita funkce uspávání zařízení (hluboký spánek, en: deep sleep), která je v modulu již zabudovaná. Pro čas zařízení ve spánku jsem definoval konstantu a faktor převodu z mikrosekund na milisekundy.

```
#define uS_TO_S_FACTOR 1000000ULL  //konverzní faktor převodu
#define TIME_TO_SLEEP 900        // Čas spánku zařízení 900s = 15min
```

Listing 3.3: Soubor: ESP32_deep_sleep.ino

V režimu hlubokého spánku je vypnuto hlavní CPU (procesor), RAM (paměť) a další digitální periferie, které by jinak spotřebovávaly energii zdroje. Jenom pár částí zůstává při spánku aktivních, RTC kontrolér, RTC periférie a RTC paměti. [24] Díky tomu spotřeba během hlubokého spánku může klesnout až na 10 mikroampér, a proto zařízení vydrží podstatně déle fungovat (pro srovnání, běžná spotřeba u tohoto zařízení při práci s Wifi a aktivním procesorem je zhruba 120 miliampér tedy 12 000x více).

Režimů spánku má ESP32 více, těmi ale nebylo třeba se zabývat, protože bylo nutné udržet jenom pár hodnot v paměti během spánku zařízení, a proto byl vhodný tento režim spánku se zapnutými RTC periferiemi.



Obrázek 3.7: Aktivní a neaktivní moduly během hlubokém spánku ESP32 [24]

Po uplynutí doby TIME_TO_SLEEP se zařízení vzbudí a začne vykonávat program od začátku. Tímto se ale všechny vytvořené proměnné v předchozí iteraci cyklu resetují na původní hodnotu. Pokud je nutné po dobu spánku proměnné zachovat, je třeba je uložit do speciálního místa v paměti (RTC memory), kde se udržují.

```
RTC_DATA_ATTR int bootCount = 0;
// proměnná pro uložení počtu probuzení zařízení ze spánku (udává počet iterací
    cyklu)
```

Listing 3.4: Soubor: ESP32_deep_sleep.ino

Poslední proměnná zakládaná před startem hlavní funkce programu byla URL adresa lokálního serveru (v mém případě IP adresa), přes kterou se bude program připojovat na web server, kam se budou přečtená data posílat.

```
const char* host = "192.168.0.202";
```

Listing 3.5: Soubor: ESP32_deep_sleep.ino

Hlavní funkce programu

Aby nahrání programu do zařízení ESP32 bylo úspěšné, musí obsahovat dvě hlavní funkce – *setup()* a *loop()*.

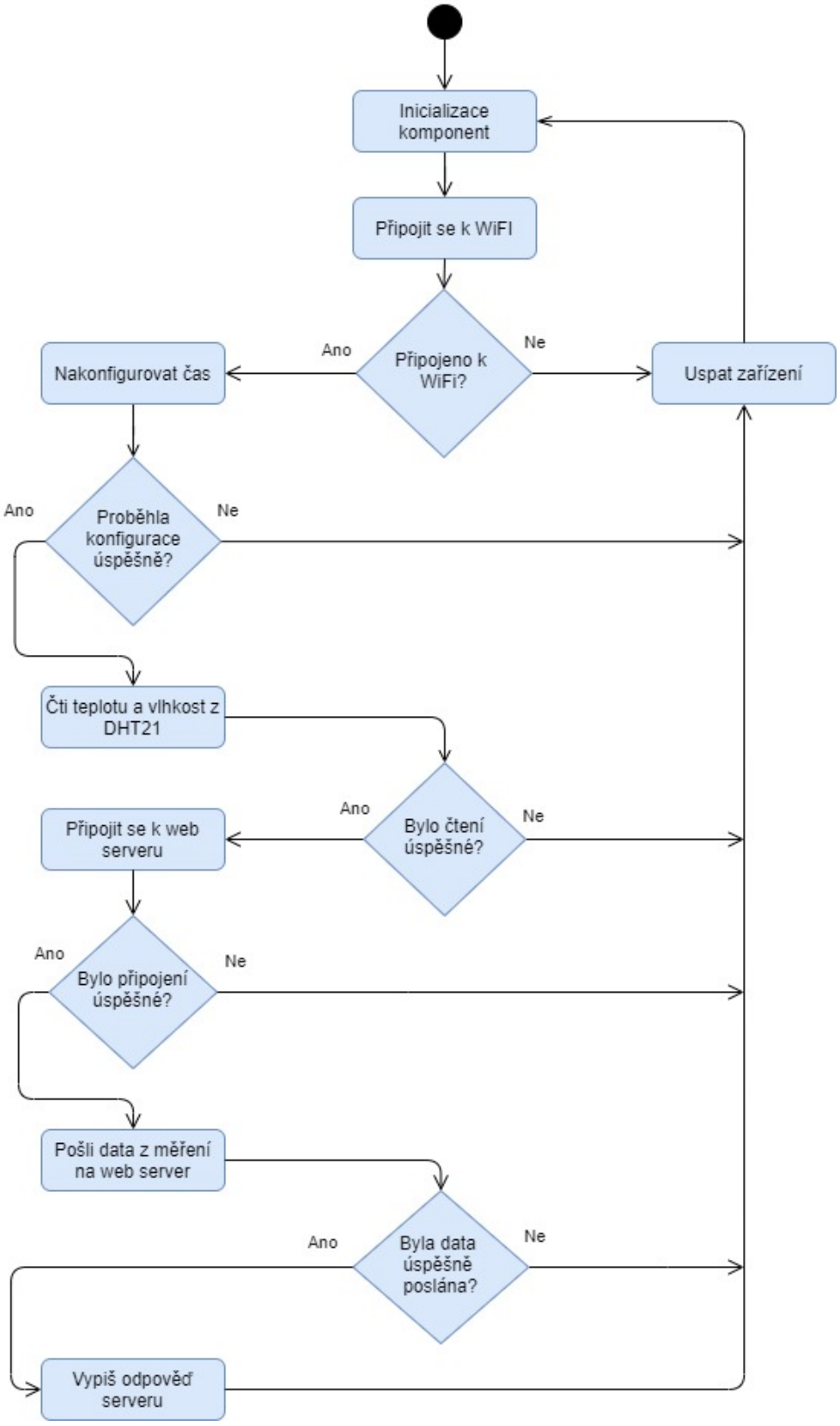
Funkce *setup()* slouží pro inicializaci základních stavebních jednotek programu (definování pinů, vytvoření objektu pro práci s Wifi nebo zahájení komunikace po sériové lince). Tato funkce se spustí jen jednou při startu zařízení.

Funkce *loop()* se spouští po funkci *setup()* a slouží jako smyčka v programu, kde je vykonávána hlavní pracovní část celé aplikace. Volá se opětovně dokola (dále budu tento proces označovat jako cyklus), dokud zařízení není vypnuto nebo uspáno.

```
void setup() {
    // zde se spustí veškerý inicializační kód
}
void loop() {
  // zde se vloží kód, který chceme volat opakovaně
}
```

V mém systému ale nikdy nebude potřeba volat funkci *loop()* z důvodu ukládání zařízení do režimu spánku. Pokud se zařízení vzbudí, vykoná pouze jednu iteraci cyklu čtení, připojení na server a odeslání dat. Pak se opět přepne do režimu spánku bez nutnosti provádění funkce *loop()*. Cyklus programu tedy vznikne opakovaným probouzením zařízení a ukládáním do spánku ve funkci *setup()*.

Vývojový diagram programu je zobrazen na obrázku 3.8. Tento diagram představuje posloupnost kroků ve funkci setup(). Program začíná inicializací komponent, spuštění sériové linky a přihlášení k Wifi. Pokud hned na začátku těchto akcí (nebo i dále v programu) nastane chyba (výpadek Wifi, nemožnost připojení k sériové lince, atp.), uloží se zařízení ke spánku, a po probuzení se program pokusí projít celý tento proces znovu.



Obrázek 3.8: Vývojový diagram programu pro záznam teploty a vlhkosti

Inicializace komponent

V první fázi funkce • je zahájena komunikace přes sériovou linku, která slouží k interakci mezi uživatelem a mikrokontrolérem. Pokud nastane nějaká chyba při vykonávání kódu, pomocí sériové linky je možné si poslat informaci, k jaké chybě došlo nebo zobrazit průběžné informace o stavu programu. Tento způsob ladění programu se využívá při připojení k počítači a ve finálním řešení nemusí být vůbec obsažen.

Po startu se na sériový monitor vypíše číslo probuzení (také číslo iterace) a důvod probuzení zařízení (v tomto systému se bude jednat vždy o probuzení pomocí vnitřního časovače).

```
Serial.begin(9600); // spuštění sériové linky, spuštěno s rychlostí 9600 Baudů (
    jednotka modulační rychlosti přenosového média za jednu sekundu)
bootCount++;
Serial.println("Boot number: " + String(bootCount));
print_wakeup_reason();
```

Listing 3.6: Soubor: ESP32_deep_sleep.ino

Připojení k Wifi

Dále je v programu velmi důležité připojení ESP32 k lokální Wifi. Pokud se aplikace nepřipojí, nemá kam data odeslat a nezjistí aktuální čas, který je potřebný ke kompletaci ukládaných dat. K inicializaci Wifi slouží funkce *WiFi.begin(ssid, password)*, která jako parametr přebírá platné přihlašovací údaje a pokusí se k Wifi připojit. Připojování má maximální limit pěti sekund, aby nedošlo k nekonečnému prázdnému zacyklení programu. Pokud se k Wifi do té doby k zařízení nepřipojí, je program ukončen a zařízení uspáno.

```
unsigned long timeout = millis(); // získání aktuálního mezičasu v programu
while (WiFi.status() != WL_CONNECTED) {
   if(millis() - timeout > 5000){
      Serial.println("Connection to wifi failed!");
      goToSleep();
   }
   else{
      delay(500);     // po 500 milisekundách se vypíše tečka pro informaci, že se
          zařízení snaží stále připojit
      Serial.print(".");
   }
}
```

Listing 3.7: Soubor: ESP32_deep_sleep.ino

Konfigurace času a inicializace senzoru

Po úspěšném připojení k Wifi se musí zařízení připojit k NTP serveru a uložit si aktuální čas. Pokud se připojení nepodaří, zařízení je opět uspáno. Pokud se připojení podaří, zařízení pošle po sériové lince aktuální čas a inicializuje senzor DHT21 pro měření teploty a vlhkosti.

```
configTime(gmtOffset_sec, daylightOffset_sec, ntpServer); //inicializace času
String myTime = getActLocalTime();
if(myTime == ""){
   Serial.println(F("Failed to read time!"));
   goToSleep();
}
printLocalTime(); // zobrazení času
dht.begin(); // inicializace DHT21 senzoru
```

Listing 3.8: Soubor: ESP32_deep_sleep.ino

Nyní je zařízení připraveno data přečíst. Na vytvořeném objektu dht, se zavolají metody *readHumidity()* a *readTemperature()*. Které jako návratovou hodnotu vrací data poslaná senzorem. Pokud senzor pošle chybné hodnoty, je program uveden do režimu spánku.

```
float h = dht.readHumidity(); // čtení vlhkosti
float t = dht.readTemperature(); // čtení teploty
if (isnan(h) || isnan(t)) {
   Serial.println(F("Failed to read from DHT sensor!"));
   goToSleep();
}
```

Listing 3.9: Soubor: ESP32_deep_sleep.ino

Připojení k web serveru

Pro připojení k web serveru je definovaná třída *WiFiClient* v knihovně WiFi.h. Lze díky ní vytvořit objekt s označením „klient“, na kterém lze zavolat metodu *connect(host, port)* k navázání spojení se serverem přes HTTP protokol. Pokud se spojení nepodaří, je zařízení opět uspáno.

```
WiFiClient client;
const int httpPort = 80; // implicitní port pro protokol HTTP

if (!client.connect(host, httpPort)){ // hostem je IP adresa serveru 192.168.0.202
   Serial.println("connection failed");
```

```
  goToSleep();
}
```

Listing 3.10: Soubor: ESP32_deep_sleep.ino

Tvorba požadavku na web server (poslání dat)

Proces práce serveru s požadavky bude více popsán v kapitole 4. Pro tento případ v programu bude použit požadavek Get. Pomocí požadavku Get lze poslat na web server data, která jsou v aktuální iteraci cyklu uložena v paměti zařízení a lze je dále uložit do databáze na web serveru. Před posláním je třeba definovat cestu k PHP souboru, který na web serveru data zpracuje a uloží do databáze.

```
//cesta k PHP souboru, který data zpracuje a uloží do databáze
String url = "/php/temphum/insert.php?";

url = url + "temperature=" + t + "&";
url = url + "humidity=" + h + "&";
url = url + "dayTime=" + myTime;
client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                    "Host: " + host + "\r\n" +
                    "Authorization: Basic amVycnk6MTIzNA==" + "\r\n" +
                    "Connection: keep-alive\r\n\r\n");
```

Listing 3.11: Soubor: ESP32_deep_sleep.ino

Do metody client.print() je v parametru předán celý Get požadavek. Tento požadavek obsahuje adresu k PHP souboru, posílaná data (teplota, vlhkost, čas), adresu serveru a platné šifrované autorizační údaje. Po přijetí požadavku web server ověří a zjistí, zda má zařízení platný přístup k serveru a jeho obsahu.

V těle požadavku je také obsaženo klíčové slovo „Connection“ které determinuje, jestli se spojení se serverem ukončí, nebo udrží (např. pro zaslání dalších dat nebo čtení odpovědi). V tomto případě je třeba spojení ponechat aktivní „keep-alive“ a čekat na potvrzení serveru, že data byla správně odeslána a přijata. Server díky tomu neukončí spojení a vrátí odpověď o přijetí/nepřijetí požadavku ve formátu JSON.

Výsledný Get požadavek na web server by mohl vypadat takto:
GET /php/temphum/insert.php? temperature=22.20&humidity=55.20
& dayTime=2021-02-04T15:13:58Z HTTP/1.1 Host: 192.168.0.202
Authorization: Basic asKlkyk6MLSzNV== Connection: keep-alive

Udržení spojení s web serverem (čtení příchozích dat)

Po odeslání dat je spuštěn cyklus, který ověří aktivnost spojení s web serverem a naslouchá (dokud je spojení aktivní), jestli server posílá zpátky data o přijetí/nepřijetí poslaných dat. Po přečtení všech dostupných dat je cyklus ukončen a z přijatých dat je možné se dozvědět, jestli byla data z monitoringu úspěšně odeslána a přijata.

```
while (client.available() == 0) { // ověření, jestli je spojení dostupné
   if (millis() - timeout > 5000) {
      Serial.println(">>> Client Timeout !");
      client.stop();
      goToSleep(); // pokud ne, uspi zařízení
   }
}
while(client.available()) { // dokud je spojení dostupné, čti data
   String line = client.readStringUntil('\r');
   Serial.print(line);
 }
goToSleep();
```

Listing 3.12: Soubor: ESP32_deep_sleep.ino

Ukládání zařízení do hlubokého spánku

Vždy na konci programu je zavolána funkce pro uspání zařízení a po jejím vykonání je celá iterace cyklu dokončena. Pro definici času stráveného ve spánku je spuštěna funkce *esp_sleep_enable_timer_wakeup()*, které je předán v sekundách čas spánku vynásobený konverzním faktorem pro jejich převedení na mikrosekundy. K uložení ESP32 do spánku slouží funkce *esp_deep_sleep_start()*, která ukončí vykonávání dalšího kódu a vypne nepotřebné periferie zařízení.

```
Serial.println("Going to sleep now for " + String(TIME_TO_SLEEP) + " Seconds");
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
esp_deep_sleep_start(); // kód za touto funkcí se nikdy neprovede
```

Listing 3.13: Tělo funkce: goToSleep()

V této kapitole měl být také popsán import dat z předpovědního portálu. Toto téma popíši až v kapitole 4, jelikož se toto téma přímo váže na operační systém serveru, který budu v mém systému používat.

Kapitola 4

Záznam naměřených a získaných dat na vlastním serveru

Data, která senzory pravidelně posílají, je nutné ukládat na paměťovém médiu, aby se dala následně vizualizovat a aby mohl podle nich systém (např. web server) spouštět další funkce systému. Data lze ukládat na různá paměťová média podle rozsáhlosti systému.

Příklady paměťových médií pro ukládání dat a práci s daty

1. **CD/DVD** - Ukládání na CD/DVD je jedno z nejlevnějších a nejdostupnějších řešení. Data lze vypálit na datový nosič. Nevýhodou je, že CD a DVD mají omezenou kapacitu, jsou náchylné na poškození a nelze na ně zapisovat opakovaně (mimo CD-RW disky). Pro ukládání velkého množství dat, to tedy není vhodné řešení. V mém systému jsem tato úložiště nepoužil.
2. **HDD/SDD disk** – Jedná se o lepší možnost ukládání než CD nebo DVD. HDD i SSD disky mají násobně vyšší kapacitu, lze na ně zapisovat opakovaně a také déle vydrží. V mém systému budou dále použity jako součást síťového úložiště a budou se zde ukládat zaznamenaná data ze senzorů.
3. **Flash disk / SD karta** – Tato úložiště jsou podobná zálohovací média jako externí pevný disk, jen obvykle s nižší kapacitou a nižší odolností. Mají malé rozměry a lze je tak pomocí dalších rozhraní připojit např. k mikrokontroléru. V mém systému by mohly sloužit jako dočasná úložiště pro data, která z mikrokontroléru nemohla být zapsána na server kvůli chybě, nebo nedostupnosti internetu. Jako finální řešení však vhodná nejsou kvůli malé velikosti úložiště a menší odolnosti. V mém systému jsem tato úložiště nepoužil.

4.1 Síťová úložiště pro ukládání a správu dat

Pro ukládání dat na paměťová média je třeba použít zařízení, které data předtím zvládne přijmout a zpracovat. Kromě fyzického zařízení s výpočetním výkonem je potřebné zajistit i software pro práci s daty. V mém systému se pro správu dat využívá software web server (software i hardware web server bude popsán v textu dále). Tento software web server může např. fungovat na lokálním NAS serveru nebo serveru cloudovém.

1. **NAS servery** - Síťová úložiště, do kterých lze vložit jeden nebo více pevných disků. Lze na nich spustit technologii RAID, která zajišťuje spolehlivé ukládání dat, protože se data zapisují na více disků současně, nebo obsahují prostředky pro zpětné získání dat (podle typu technologie RAID). Při poškození nebo výpadku disku tak je možnost data obnovit. NAS server je obvykle připojený k internetové síti a lze k němu přistupovat z lokální sítě i z internetu. Podobá se proto soukromému cloudu. Pro ukládání dat v mém systému je tato možnost vhodná z důvodu možnosti navýšení úložné kapacity, rychlého a bezpečného přístupu k datům a obsahu dalších funkcí pro práci s daty. Lze jej provozovat jako plnohodnotný server a spolehlivé úložiště.
2. **Cloudový server** – Cloudový server slouží k ukládání dat přes internet. Tuto možnost ukládání zprostředkovávají internetoví poskytovatelé např. Google, Microsoft nebo Dropbox. U cloudů je velká výhoda, že uživatelská data jsou uložena (a roztroušena) v datových centrech po celém světě. Jsou tak velmi dobře zabezpečená a lze se k nim dostat odkudkoliv z internetu. Pro ukládání dat v mém systému tato možnost byla zvolena jen v rámci testování. Jako úložný prostor je tento typ úložiště vhodný. Nevýhodou je nižší rychlost oproti lokálnímu NAS serveru a nemožnost zasahovat do hardware/software vybavení serveru. Další důvod je vysoká cena, která je pro 1TB dat průměrně 200 Kč měsíčně. Dlouhodobě toto řešení tedy není pro můj systém vhodné.

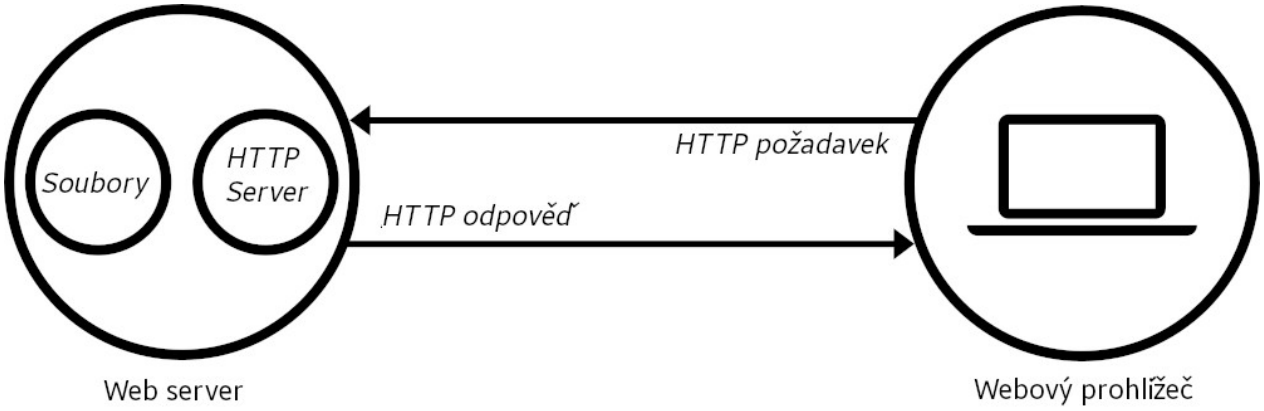
Web server

Před popisem jednotlivých síťových úložišť je nutné vymezit pojem web server. Pojem web server může odkazovat na hardware, software nebo na kombinaci obou dvou.

Hardware web server je počítač, na kterém běží webový software a je uložen jeho souborový obsah (HTML, CSS dokumenty, obrázky, JavaScript soubory). Hardware web server je připojený k internetu a slouží k vyměňování fyzických dat s jinými zařízeními připojenými k internetu. Jako každý jiný má i tento počítač operační systém (Linux, Windows, Mac, atd.).

Software web server na druhou stranu obsahuje části, které kontrolují, jak webový uživatel přistupuje k serverovým datům. Jako základ se používají HTTP servery. HTTP server je software, který umí pracovat s URL (webové adresy) a HTTP protokolem (využívaným pro zobrazování webových stránek uživateli). K HTTP serveru lze přistupovat prostřednictvím doménových jmen

webových stránek, které server obsahuje a poskytuje tak obsah těchto hostovaných webů do koncového zařízení uživatele. [25]



Obrázek 4.1: Ukázka základní HTTP komunikace

Na nejnižší úrovni kdykoliv webový prohlížeč potřebuje soubor, který je hostován na serveru, prohlížeč pošle požadavek na tento soubor skrze HTTP. Když požadavek dorazí na (hardware) web server, (software) web server přijme požadavek, najde požadovaný soubor a pošle ho zpět prohlížeči také skrze HTTP. Pokud (software) web server nenajde požadovaný soubor, vrátí chybovou hlášku.

Jeden z nejpopulárnějších software web serverů je Apache, který bude v mém systému použit. Mezi další populární servery patří Nginx nebo Microsoft-IIS.

Technologie web serveru

Aby mohl software web server dobře pracovat a zobrazovat správně obsah, musí mít k tomu potřebné nástroje. Pro většinu webových stránek je třeba, aby systém uměl pracovat s databází nebo aby průběžně posílal data uživateli. K naprogramování funkcionality se používají web serverové technologie MariaDB, MySQL a jazyky jako SQL, C#, PHP, Java nebo Python. V mém systému jsou použity technologie PHP a MariaDB, jelikož jsou dobře optimalizované pro spolupráci mezi sebou a s webovým serverem Apache. Tato technologická trojice pod zkratkou AMP (Apache, MariaDB, PHP) je společně s operačním systémem Linux jednou z nejrozšířenějších kombinací mezi webovými servery (označení LAMP). Písmenem M ve zkratce LAMP může být označen také databázový systém MySQL, který je ale v poslední době často nahrazen jeho vylepšením a nástupcem, technologií MariaDB. Mezi další možnosti provozování web serveru patří např. technologie WNMP (Windows, Ngix, MySQL, PHP).

Do mého systému byly vybrány technologie AMP, protože jsou dostupné na většině cloudových web serverů, a také na mém domácím NAS serveru, který byl použit v mém systému. Kromě dostupnosti jsou také velmi rychlé, spolehlivé a dobře pracují mezi sebou a s operačním systémem

Linux, který je instalován i na mém lokálním NAS serveru. Tyto technologie a funkčnost systému pro záznam dat byly v mém systému vyzkoušeny na lokálním NAS serveru i serveru cloudovém.

4.2 Cloudový server – webhosting

Jako první zkušební řešení byla v mém systému pro ukládání dat a funkce web serveru využita platforma poskytovatele služeb webhostingu webzdarma.cz. Pojem webhosting znamená pronájem prostoru pro webové stránky (a dalších funkcí jako e-mail, databáze, atd.). Na internetu poskytuje webhosting velké množství poskytovatelů. Do mého systému byl zvolen poskytovatel webzdarma.cz, protože podporuje technologie AMP, tedy Apache 2.4, PHP verze 7 a ukládání až 500 Mb v databázi MySQL zdarma.

Výhody:

- Uživatel se nemusí starat o hardware zařízení, na kterých web server běží. V případě nefunkčnosti např. z důvodu výpadku serveru má zodpovědnost poskytovatel.
- Je to velmi bezpečné řešení. Poskytovatelé pečlivě zabezpečují své servery kvůli potenciálním útokům a unikům uživatelských dat.
- Základní funkce a prostor jsou zdarma. Uživatel si tak může otestovat základní funkčnost svých webových stránek.

Nevýhody:

- Větší prostor a další funkce jsou zpoplatněny. Pokud přibyde dat v databázi, musí si uživatel zakoupit lepší balík funkcí za více peněz.
- Uživatel nemá kontrolu nad obsahem a správou technologií. Pokud by uživatel chtěl používat nový modul/balík pro jazyk PHP, musí pro instalaci kontaktovat administrátora webhostingu. Tyto úkony jsou také zpoplatněny.

4.3 Lokální (domácí) NAS server

Další způsob správy a ukládání dat na web serveru je možné vytvořením domácího serveru např. ze starého počítače nebo koupí vlastního serveru k tomuto účelu uzpůsobenému. V mém systému bylo využito domácího Synology NAS serveru. Jedná se o počítač s operačním systémem založeným na Linuxu a jeho primární účel je správa a záloha dat např. v domácí či firemní síti. Obsahuje ale spoustu dalších funkcí. Dá se využít jako DNS, DHCP, VPN nebo funkční web server. Systém také umožňuje připojení přes SSH. Administrátor díky této funkci může pracovat i ze vzdáleného počítače pomocí vzdáleného terminálu. Systém také dovoluje spouštění vlastních skriptů a plánování těchto úloh. Této funkčnosti plánování a spouštění skriptů využiji v mém systému ke čtení dat z předpovědního portálu a k jisté formě automatizace.

Výhody:

- Uživatel má plnou kontrolu nad balíky funkcí, které server/web server potřebuje.
- Lze nastavit automatické aktualizace systému. Vždy je tedy systém aktuální bez nutnosti zásahu uživatele. Lze také nastavit automatické zálohování systému.
- Poskytuje četnou řadu funkcí navíc pro práci v lokální síti (např. DNS, DHCP, VPN, Web server, Media server, File station, atd.)
- Podporuje i další funkce jako např. RAID, antivirus, sledovací stanice, WordPress nebo Git server
- Poskytuje více možností pro jeho připojení z/do internetu.

Nevýhody:

- Vyšší pořizovací cena (okolo 10 000 Kč za systém s 1TB úložištěm). Z dlouhodobého hlediska je však tato možnost výhodná ve srovnání s webhostingem z důvodu vysokých měsíčních poplatků za funkce i prostor.
- Nižší bezpečnost oproti webhostingu, pokud uživatel neaktualizuje systémové balíky nebo pokud není správně nastaven firewall a další bezpečnostní funkce.

V mém systému byl použit server Synology DiskStation DS712+ obshaující dva 1TB disky. Toto zařízení bylo využito z důvodu nevyužité kapacity tohoto zařízení v původní lokální síti budovy a z ekonomických důvodů pro ušetření dalších nákladů pro provoz budovy.

Obrázek 4.2: Synology DiskStation DS712+

Pro funkci mého systému byly na tento server instalovány komponenty pro web server Apache 2.4, PHP 7.2 a MariaDB. Pro jednodušší práci s databází byl na server Synology také instalován programový systém phpMyAdmin.

4.3.1 Připojení serveru do veřejné části internetu

V průběhu vytváření bylo zařízení zapojeno technologií Ethernet jen do vnitřní lokální sítě, ale web server měl být také zpřístupněn i veřejné části internetu. Zde nastal problém, protože NAS v této domácí síti, nešel jednoduše bezpečně připojit do celosvětové sítě internetu. Pro tento účel poskytuje Synology několik možností.

První možnost je mít na lokálním směrovači přidělenou pevnou (statickou) IP adresu od poskytovatele internetu. Bohužel i náš poskytovatel jako většina jiných, adresu přiřazuje dynamicky. Toto řešení se statickou adresou je však cenově nákladné (okolo 50 Kč měsíčně - firma KlimNet z.s.). Z ekonomických důvodů tak byla vybrána jiná varianta.

Druhá možnost je vyhrazení speciálního portu na ISP směrovači poskytovatele, přes který se dále směrují požadavky protokolu HTTP na náš domácí směrovač. Této možnosti provozu se říká přesměrování portů (en: port forwarding). Provoz lze pak dále směrovat z domácího směrovače na lokální IP adresu serveru NAS. Tato možnost připojení ale přestane fungovat ve chvíli, kdy nám poskytovatel změní naší dynamickou IP adresu. Tento problém se dá vyřešit pomocí služby DDNS od Synology, kterou popíši dále.

V tuto chvíli vývoje již bylo možné se na server připojit odkudkoliv z internetu skrze tuto dynamickou IP adresu a přiřazený port – 213.143.45.22:10936. Pokud by ale poskytovatel naši IP adresu změnil, přestalo by připojení fungovat. Software od Synology je pro tento problém připraven se službou DDNS. Tato služba nabízí připojení k zařízení Synology NAS přes internet prostřednictvím mapování názvu hostitele k IP adrese, která může být dynamická, jelikož se služba DDNS automaticky stará o mapování aktuální (správné) IP adresy na doménové jméno. Tato služba pro svou funkcionalitu tedy potřebuje funkční doménové jméno, které se mapuje na měnící se IP adresu poskytovatele internetu. Firma Synology je také mimo jiné poskytovatelem doménových jmen. Doména tedy byla rezervována u společnosti Synology z důvodu možnosti správy domény na NAS serveru - www.iotcontrol.synology.me.

Po zprovoznění služby DDNS se tak mohl kdokoliv a odkudkoliv připojit na webový server s adresou a portem www.iotcontrol.synology.me:10936. Bohužel tato doména fungovala pouze mimo domácí síť a při záznamu dat z domácích senzorů byl systém stále závislý na vnitřní IP adrese serveru NAS.

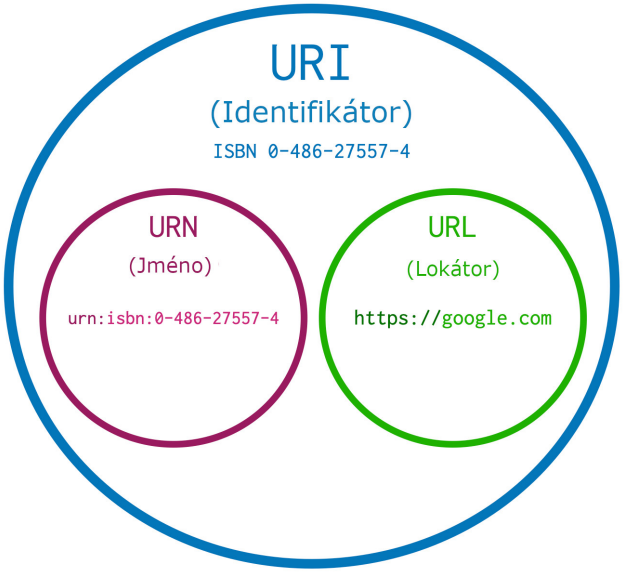
Aby ale nebylo nutné si číselnou IP adresu serveru NAS pamatovat, lze ji převést na doménové jméno pomocí DNS serveru. Pro tento účel byl opět využit server DS712+, jelikož tuto funkci přímo podporuje a lze ji jednoduše nastavit. Po vytvoření DNS serveru byl také vytvořen nový záznam shodný s doménovým jménem vytvořeným u společnosti Synology (www.iotcontrol.synology.me).

Doménová jména mohou být v tomto případě stejná, pokud je v lokální síti nastaven DNS server se zvoleným záznamem jako hlavní, jelikož se při překladu jmen na IP adresy mapuje prvně z hlavního serveru. V posledním kroku bylo tedy nutné nastavit vytvořený DNS server jako hlavní DNS server sítě pro ukládání doménových jmen.

4.4 Program pro ukládání dat na web serveru

V kapitole 3 bylo zmíněno, že software web server pracuje na principu přijímání a odesílání požadavků pomocí protokolu HTTP. Těchto požadavků je více typů a každý se používá pro jiný přenos dat (např. Put, Post, Delete nebo Get). Pro odesílání dat z mikrokontroléru NodeMCU-S ESP32 i přijímání dat na koncových zařízeních bylo využito požadavku Get, který pošle na server zaznamenaná data a vrátí odpověď ze serveru o úspěšnosti přenosu. Tyto požadavky jsou směrovány na web server na základě URI adresy.

Adresa URI je řetězec znaků používaných pro identifikaci zdroje na internetu. Může identifikovat zdroj podle cesty, jména nebo kombinací obou. Dělí se na dva typy - URL a URN. URN je podmnožina URI, která specifikuje, kde je zdroj uložen a jak se k němu dostat (http://, ftp:// nebo smb://). URN je pak podmnožina URI, která v sobě drží informaci o názvu souboru, ale neobsahuje informaci, jestli je zdroj dostupný.



Obrázek 4.3: Ukázka rozdílu mezi URL, URI a URN [26]

Pokud by systém chtěl načíst hlavní stránku webového serveru z vnitřní sítě, vypadala by URI (konkrétně tedy URL) takto - *http://192.168.0.202/index.html*

Ukládání dat na web serveru

Když na web server přijde požadavek z uživatelského prohlížeče, HTTP server Apache jej vyzvedne a pošle ho ke zpracování jazyku PHP. PHP se podle potřeby připojí k databázi a vykoná potřebné SQL dotazy pro vrácení dat. Jednoduchý požadavek může vypadat následovně:

http://192.168.0.202/php/temphum/insert.php?temperature=22&humidity=55

Modrá část obsahuje cestu k souboru, který se má na serveru vykonat a červená část obsahuje data k poslání na server. Server v tomto případě spustí vykonávání kódu v souboru „insert.php“. První krok pro uložení dat je připojení k databázi, kde budou data uložena do příslušné tabulky. Pokud připojení neproběhne úspěšně, je program ukončen a zobrazí chybovou hlášku.

```
$mysqli = new mysqli("nazev_serveru", "uzivatel", "heslo", "nazev_databaze");
if ($mysqli->connect_errno) {
   printf("Connect failed: %s\n", $mysqli->connect_error);
   exit(); // pokud připojení není úspěšné, ukončí další vykonávání kódu
}
```

Listing 4.1: Soubor: connect_db.php

Po úspěšném připojení se ověří, jestli hodnoty měření předané v URI jsou všechny, pokud ano, uloží se do pomocných lokálních proměnných, které se následně vloží do SQL dotazu na databázi.

```
if (isset($_GET['temperature']) && isset($_GET['humidity']) &&
    isset($_GET['dayTime'])) {

   $temperature= $_GET['temperature'];
   $humidity= $_GET['humidity'];
   $dayTime= $_GET['dayTime'];

// dotaz pro vložení teploty, vlhkosti a času do databáze
$result = $mysqli->query(
   "INSERT INTO temphum(temperature, humidity, dayTime)
    VALUES ('$temperature', '$humidity', '$dayTime')
");
```

Listing 4.2: Soubor: insert.php

Kromě teploty a vlhkosti je ukládáno také datum měření. Datum slouží pro vyobrazení veličin v reálném čase. V kódu je také podmínka, která hlídá, jestli serveru jsou posílány všechny hodnoty

pro uložení kompletního záznamu. Pokud nejsou všechny parametry pro záznam dodány (teplota, vlhkost, čas), server vrátí chybovou zprávu.

Pokud nastane nějaká chyba při nahrávání (např. do buňky pro datový typ Integer chceme uložit datový typ String), server vrátí jako odpověď chybovou zprávu. Pokud vše proběhne správně, server vrátí odpověď, že data byla úspěšně uložena.

```
if ($result === TRUE) { // dotaz proběhl úspěšně
      $response["success"] = 1;
      $response["message"] = "Temphum Data successfully inserted.";
      echo json_encode($response); // zašiforvání do formátu JSON
  } else {
      echo "Error: " . $result . "<br>" . $mysqli->error;
}
```

Listing 4.3: Soubor: insert.php

Pokud jsou všechny podmínky přenosu a uložení dodrženy, data jsou zapsána v databázi a můžeme s nimi dále pracovat. Na stejném principu funguje i čtení dat a jejich aktualizace (podle parametrů měření v URI se vrátí výsledek dotazu na databázi ve formátu JSON). S tímto principem odesílání dat je ale velmi zásadní problém se zabezpečením. Do prohlížeče může každý uživatel s připojením k internetu zadat výše zmíněnou URI a nahrát falešná data nebo v nejhorším případě napadnout celou databázi skrze SQL injekci. Těmto problémům se dá zamezit základní uživatelskou autorizací.

Autorizace zařízení

Podstatou autorizace je ověřit, zda daný klient (typicky webový prohlížeč) má oprávnění provést příslušnou akci, např. vložit nový záznam do seznamu teplot. Tohoto se dá jednoduše dosáhnout přidáním HTTP hlavičky pro autorizaci do Get požadavku. Hlavička putuje společně s daty k požadovanému serveru, kde se autorizační údaje ověří. Webový server si dodané údaje zkontroluje a na jejich základě udělí přístup. Pokud klient nepošle v prvním požadavku autorizační údaje, je serverem vyzván pomocí protokolu HTTP, aby poslal v rámci opakovaného požadavku na stránku také autentizační informace tj. jméno a heslo.

Výsledný řetězec jména a hesla je před odesláním zakódován metodou **Base64** které převádí binární data na posloupnosti tisknutelných znaků (toto kódování se také např. využívá v rozšíření MIME pro přenos emailů) aby znesnadnil čtení autentizačních informací člověkem. Není tedy jeho smyslem tyto informace kryptograficky zabezpečit. Pokud se potenciální útočník zvládne napojit na tento tok dat, data si zobrazí a podle potřeby také rozklíčuje. Přestože data nejsou více šifrovaná,

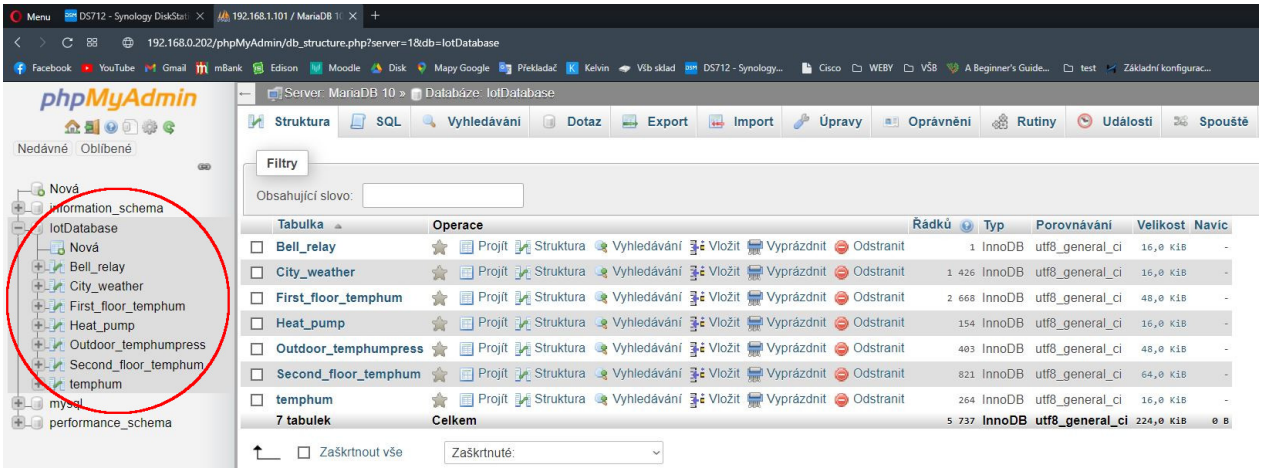
byla zvolena tato základní autorizace z důvodu, že jde o univerzální a standardizovaný mechanismus, který je dostupný na všech platformách, a protože v mém systému neposílám žádná citlivá data.

```
$valid_passwords = array ("uziv_jmeno" => "uziv_heslo");
$valid_users = array_keys($valid_passwords);
$user = $_SERVER['PHP_AUTH_USER'];
$pass = $_SERVER['PHP_AUTH_PW'];
$validated = (in_array($user, $valid_users)) &&
            ($pass == $valid_passwords[$user]);
//pokud selže autorizace, vypíše se uživateli chybová stránka
if (!$validated) { header('WWW-Authenticate: Basic realm="My Realm"');
  header('HTTP/1.0 401 Unauthorized');
  die ("Not authorized");
}
```

Listing 4.4: Soubor: req_login.php

4.4.1 Databáze pro záznam dat

Jako rozhraní pro práci s databází MariaDB byl na server Synology NAS také instalován programový systém phpMyAdmin. Tento systém umožňuje jednoduchou správu obsahu databáze MariaDB nebo MySQL přes webové rozhraní. V současné době umožňuje vytvářet/rušit databáze, vytvářet/upravovat/rušit tabuky, provádět SQL dotazy a spravovat tabulkové klíče. Pro záznam dat z podsystémů pro monitoring byla vytvořena databáze IoTDatabase, ve které byly dále vytvořeny tabulky pro rozlišení záznamů z podsystémů. Na obrázku 4.4 lze tuto databázi vidět společně s vytvořenými tabulkami.



Obrázek 4.4: Systém phpMyAdmin pro práci s databází IoTControl

4.4.2 Formát přenosu dat mezi systémy

Data z/na web server je třeba přenášet skrze internet v uceleném formátu. Obvykle se jedná o XML nebo JSON Tyto formáty se od sebe velmi liší a to především svou velikostí. XML je oproti JSON rozsáhlý formát, který obsahuje mnoho znaků navíc pro formátování a není tak vhodný pro přenos malých dat. Oproti tomu JSON obsahuje méně znaků pro formátování a proto je i celkově rychlejší. Do mého systému byl zvolen formát JSON, protože má dobrou přenosovou rychlost(z důvodu menší velikosti) a jednodušeji se zpracovává.

Ukázka formátu JSON: *{„success“ : 1 , „data“: [„temperature“: 22, „humidity“: 55]}*

4.4.3 Import dat z předpovědního portálu

Kromě dat v budově a jejího okolí se k vyhodnocování reakce na počasí často využívá i aktuální počasí ve městě. Bylo třeba tedy do databáze denně průběžně importovat hodnoty z předpovědního portálu. Do systému byl zvolen předpovědní portál OpenWeather, který umožňuje data stahovat až 60x za minutu a je zcela zdarma. Tento portál poskytuje také možnost importovat předpověď počasí až s týdenním předstihem nebo si stáhnout historická data počasí.

Na serveru byla pro tento účel stahování vytvořena úloha v plánovači úloh (linuxový systém Cron) a vytvořen skript pro stažení a uložení dat. V plánovači bylo dále nastaveno, kdy se má úloha spouštět a jak často. V tomto případě bylo vykonávání nastaveno na každý den, co 30 minut. Skript je vytvořen pomocí nástroje Curl, který se často používá k přenosu dat po síti.

```php
$url = "http://api.openweathermap.org/data/2.5/weather?q=Klimkovice";
$ch = curl_init(); // inicializace nástroje CURL
// CURL po odeslání vrátí odpověď
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_URL, $url); // mapování URL na proměnnou CURL
$result = curl_exec($ch); // vykonání dotazu a vrácení výsledku
curl_close($ch);
```

Listing 4.5: Soubor: download_weather.php

Do proměnné „$result“ se uloží data z předpovědního portálu. V této proměnné se nachází velké množství dat (teplota, vlhkost, tlak, směr vetru, rychlost větru, vlhkost, atp.). Vybraná data z této proměnné (teplota, vlhkost, tlak) jsou poté uložena do databáze stejným způsobem jako data teploty a vlhkosti zaznamenaná na senzoru DHT21 v kapitole 3. Tímto je vykonávání skriptu dokončeno a z dat se následně mohou vyhodnocovat závěry.

Kapitola 5

Prezentace a vizualizace dat získaných měřením za provozu budovy

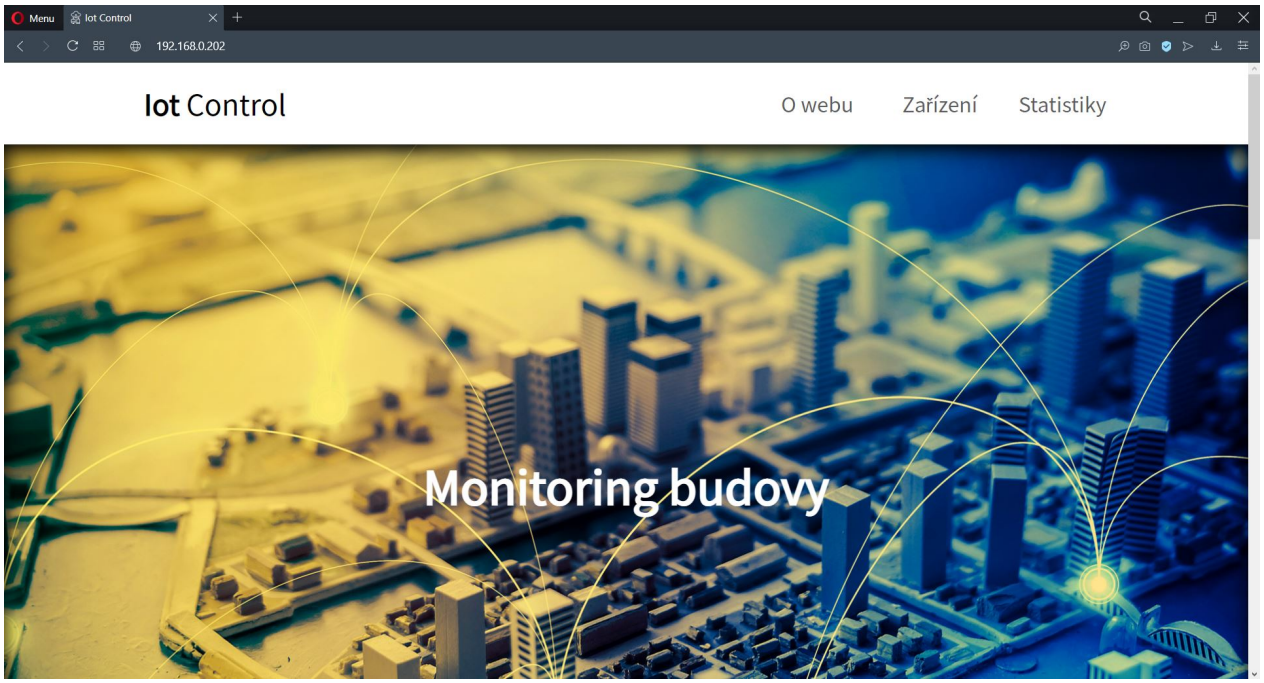
Nasbíraná data je nutné uživateli někde zobrazit. V současné době počítačů a chytrých telefonů je možné si zobrazit tyto informace odkudkoliv z chytrého zařízení. Slouží nám k tomu např. mobilní aplikace nebo webové stránky. Obě varianty byly do mého systému zahrnuty, aby byla data jednoduše zobrazitelná.

5.1 Webová stránka

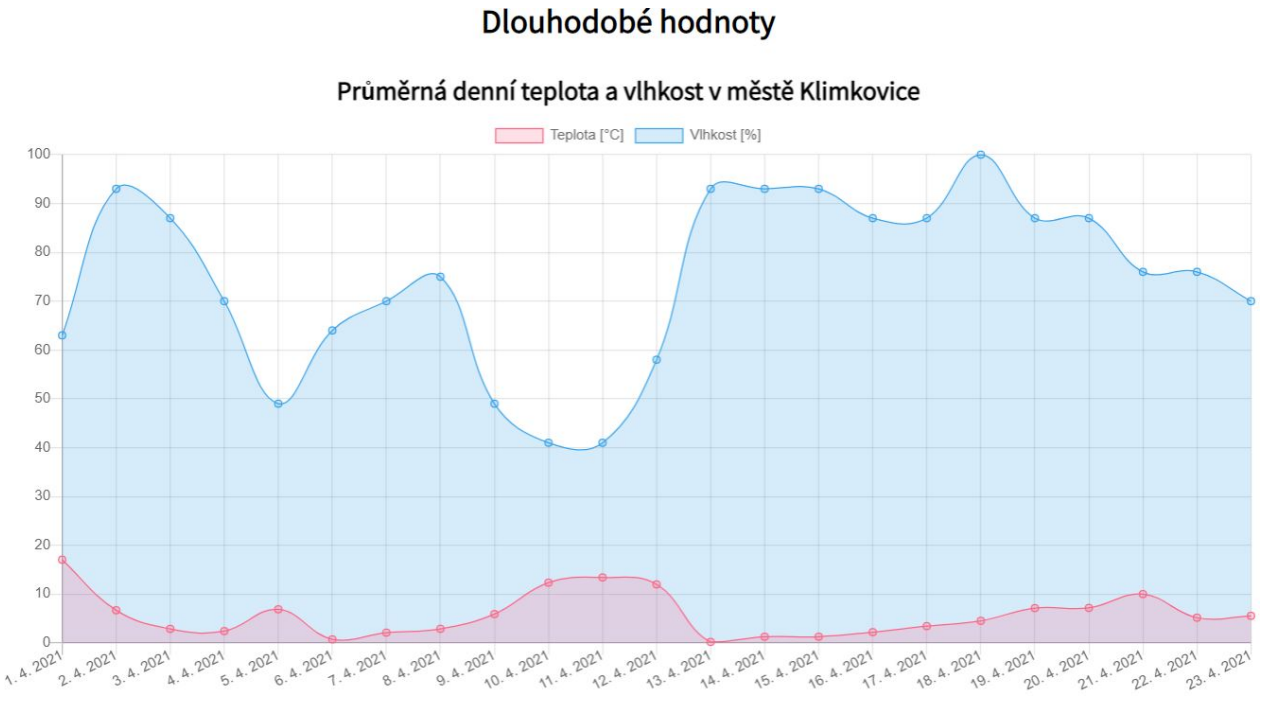
Pokud je server NAS zapnut, je pro zobrazení dat po internetu dostupná webová stránka systému. Tato stránka umožňuje prohlédnout si některé části vytvořeného systému a vizualizuje nasbíraná data ze senzorů. Webová stránka je rozdělena na tři části, hlavní stránka, stránka o zařízeních a stránka statistik.

Na hlavní stránce je popis účelu stránky a odkazy na další podstránky. Na stránce zařízení je seznam použitých zařízení, jejich informace o aktuálním stavu a popis významu v systému. Na poslední stránce - statistiky, jsou zobrazeny informace o počasí ve městě a dlouhodobých hodnotách uvnitř budovy. Ukázka hlavní stránky je zobrazena na obrázku 5.1.

Vizualizace tedy probíhá na stránce statistik, kde jsou vytvořeny grafy ze zaznamenaných dat pomocí JavaScriptové knihovy Chart.js. Tato knihovna umožňuje jednoduchou a efektivní tvorbu grafů a její příklad lze nalézt na obrázku 5.2 který zobrazuje průměrné denní teploty a vlhkost posledního měsíce ve městě Klimkovice.



Obrázek 5.1: Hlavní stránka webové aplikace zobrazena z lokální adresy 192.168.0.202

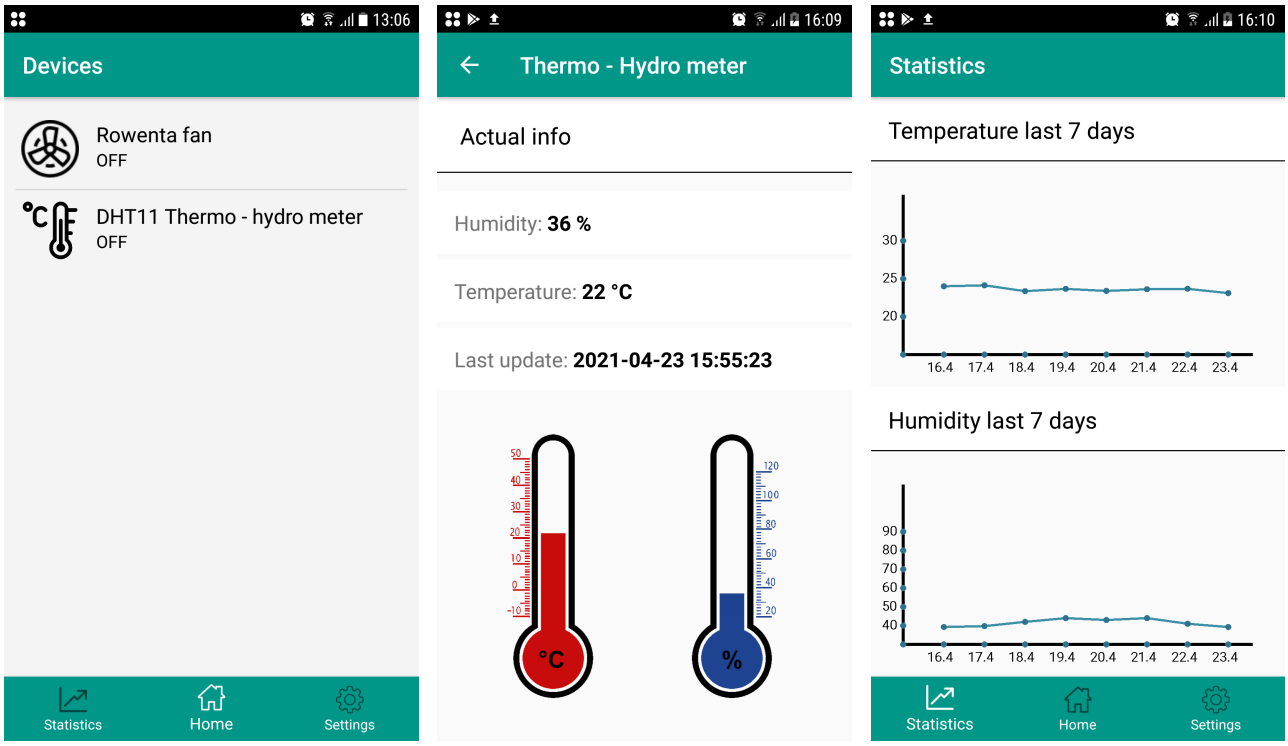


Obrázek 5.2: Ukázka grafu teploty a vlhkosti ve webové aplikaci

5.2 Aplikace pro chytré telefony

Aplikace na telefon byla vytvořena pro platformu Android, konkrétně pro verzi 7.0 a výše. Je postavená na technologii Java a poskytuje základní rozhraní pro zobrazování dat. Obsahuje fragment s malou statistikou, hlavní stránku a nastavení. Každé zařízení má pak dále svůj vlastní fragment pro zobrazení dat. Pokud uživatel aplikaci opustí, stále běží na pozadí a upozorní uživatele notifikací, pokud dojde k překročení minimální či maximální vlhkosti ve vnitřních prostorách domu.

Pro vizualizaci dat je vytvořen fragment s grafy. Grafy jsou v tomto případě dva a poskytují uživateli informaci o teplotě a vlhkosti v domě za poslední týden. Tento graf můžeme vidět na obrázku 5.3c



(a) ukázka hlavní stránky aplikace

(b) ukázka stránky zařízení pro záznam teploty a vlhkosti

(c) ukázka hlavní stránky aplikace

Obrázek 5.3: Mobilní aplikace IoT Control

Kapitola 6

Umělá inteligence a strojové učení obecně

Umělá inteligence je soubor metod, přístupů a algoritmů, sloužící k řešení velmi složitých úloh. Vědec Martin Minsky jednou pronesl, že *„... umělá inteligence je věda o vytváření strojů nebo systémů, které budou při řešení určitého úkolu užívat takového postupu, který – kdyby ho dělal člověk – bychom považovali za projev jeho inteligence.“* (1967). Jiné vysvětlení zase říká, že umělá inteligence je označení pro systémy, stroje či algoritmy vytvořené k zefektivnění provádění úkolů a usnadnění lidské práce.

Na otázky, co znamená umělá inteligence nebo kdy se dá umělá inteligence považovat za inteligentní, se pokoušel zodpovědět v minulém století Alan Turing, který také vytvořil test, jež měl determinovat, jestli vytvořený systém AI vykazuje známky inteligence. Test je velmi jednoduchý a spočívá v tom, že nezávislá osoba v jedné místnosti klade otázky a předává je do druhé místnosti, kde na ně náhodně odpoví systém AI nebo člověk. Pokud testující nedokáže rozeznat, zda komunikuje se strojem nebo s člověkem, pak tento systém AI testem prošel a můžeme ho považovat za inteligentní. Tento test byl dlouhou dobu považován za základ při posuzování inteligence strojů, avšak dnes už víme, že zdaleka nepokrývá všechny aspekty inteligentních systémů. Pro příklad lze uvést tzv. paradox čínského pokoje, se kterým přišel v roce 1980 filozof John Searl. Searl si uvědomil, že splnění Turingova testu nemusí znamenat myšlení, nebo uvědomění a ukázal, že i člověk může být např. schopen setřídit tabulku s čínskými znaky, aniž by rozuměl čínsky, nebo chápal, co znaky znamenají. Pokud totiž člověku, který čínštinu neovládá, dáme určitou znalost (např. možnost nahlédnutí do knihovny) je člověk teoreticky schopen vyhledat dostatek materiálu na to, aby nalezl výskyt dodané otázky a prostým opsáním části kontextu vytvořil smysluplnou odpověď. Vnější pozorovatel by se tak mohl domnívat, že tento člověk čínštině bez problému rozumí, přestože ve skutečnosti tomu tak není, protože tento člověk pouze mechanicky pracuje s pro něj neznámými symboly, takže by jeho práci mohl zastat i zcela nemyslící stroj.

Odvětví umělé inteligence

1. **Symbolický funkcionalismus** - Základní předmět výzkumu tohoto proudu je reprezentace znalostí a inteligentní prohledávání stavového prostoru. [27]. Typická je pro tento směr logika. Jako příklad lze uvézt např. znalostní systémy, automatické dotazování nebo plánování.
2. **Konekcionismus** - Tento proud je založen na předpokladu, že inteligence plyne ze statického propojení velkého počtu stejných fixně svázaných výpočetních jednotek. Tato myšlenka AI je inspirována lidským mozkem, kde základní jednotka je neuron. Neurony na základě hodnoty součtu vážených vstupů excitují do aktivního stavu nebo nikoliv a podle toho vyhodnotí danou situaci. Jako příklad lze uvést např. neuronové sítě.
3. **Robotický funkcionalismus (behaviorismus)** - Robotický funkcionalismus je založen na předpokladu, že zkombinováním velkého počtu specializovaných, ale neinteligentních procesů lze dosáhnout inteligentního chování [28]. Tento postup se využívá hlavně v robotice a je založen na agentních systémech.

V této bakalářské práci se budu dále zabývat expertními systémy a strojovým učením, jelikož budou tvořit hlavní teoretický základ systému pro vyhodnocování řízení budov.

6.1 Expertní systémy

Odvětví expertních systémů (ES) se řadí mezi znalostní systémy umělé inteligence. Jsou to počítačové programy založené na rozhodování experta při řešení složitých úloh. Výsledný expertní systém má za cíl dosáhnout kvality rozhodování na úrovni experta lidského. Tyto systémy se liší oproti klasickým systémům tím, že není dána přesná posloupnost kroků či algoritmů ke zpracování daných vstupů.

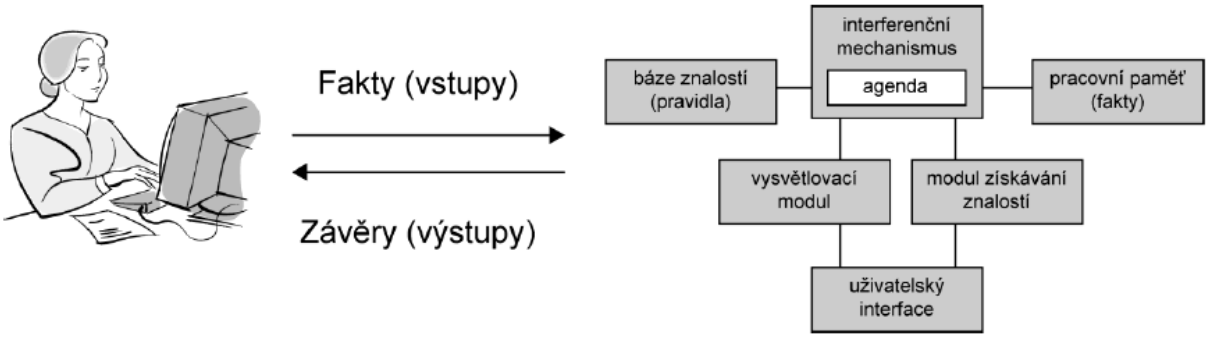
Tyto systémy jsou představiteli tzv. evolučního programování – pokud problém je analogický s problémem úspěšně vyřešeným metodou M, pak i tento problém se začne řešit metodou M. Princip spočívá v naučení rozpoznat, které zkušenosti lze použít. Inteligence tedy nespočívá v programu, ale je dána znalostmi, závěry lidských expertů, úspěšnými výpočty, které tvoří vstupy do programu. Výsledkem může být hledaný závěr, rada nebo požadavek na rozšíření znalostí lidským expertem v dané oblasti.

Mezi charakteristické rysy ES (a jejich rozlišení od klasických programů) je oddělení znalostí a mechanismu pro jejich využívání. To umožňuje vytvářet problémově nezávislé (prázdné) ES, kde jeden mechanismus může pracovat i s různými bázemi znalostí. ES musí mít také některé rysy uvažování podobné expertu lidskému. Systém stejně jako expert může pracovat s nejistými znalostmi (heuristikami) a nejistými daty (např. subjektivním pocitem člověka). Za příklad expertního systému by se dal považovat např. systém vyhodnocování nemocí. Pacientovi může být udělena otázka, jak často má teplotu. U jeho odpovědi se mohou objevovat pojmy jako „často“ nebo „někdy“ které

musíme nějak kvantifikovat. Následnou znalostí s neurčitostí může např. být „Jestliže má pacient teplotu, obvykle mu je doporučeno snížit teplotu lékem“. Pacient ale může mít často teplotu kvůli nachlazení, ale občas i v důsledku zlomeniny. Systém tedy musí umět vyhodnotit správnou příčinu a zahrnout do svých výpočtů i nejistotu a neznalost uživatele.

6.1.1 Komponenty expertních systémů

Základní představa o expertním systému je taková, že ES obsahuje **bázi znalostí, bázi dat** a **vnitřní inferenční (řídící) mechanismus**. Do ES jsou vkládány proměnné vstupy, systém provede vlastní rozhodování a vyhodnotí odpovídající závěry – výstupy. Celý tento proces a blokové schéma ES je znázorněno na obrázku 6.1.



Obrázek 6.1: Blokové schéma expertního systému [29]

Báze znalostí

V této části systému se shromažďují veškeré znalosti experta potřebné pro řešení daného problému. Je zde zapsáno velké množství různých znalostí od obecných až po odborné. Jako příklad lze uvést tvrzení - pokud student odevzdá pozdě úkol, dostane za něj špatnou známku. Pro ideální bázi znalostí je třeba zajistit, aby byla báze modulární (tj. aby bylo možné formulace bez problémů doplnit či aktualizovat). ES musí být ale také schopen využívat i nejisté znalosti, stejně jako člověk – expert. Existence těchto nepodložených, zkreslených nebo neúplných informací v bázi znalostí se označuje jako nejistota v bázi znalostí. Každému jednotlivému nejistému elementu reprezentace znalostí jsou přiřazovány numerické parametry (váhy, míry, stupně důvěry apod.), které vyjadřují míru jejich nejistoty. [30]

Každý člověk při řešení všech problémů vychází ze svých zkušeností (tzv. hloubkové znalosti). Také expertní systém musí být schopen využívat hloubkové znalosti ("používat zdravý rozum"). Každý expertní systém by měl být zkonstruován tak, aby dokázal předvídat následky všech svých činností (navržených řešení). Robotu, který má vařit špagety na plynovém vařiči, by nemělo zůstat utajeno, že vařící voda má tendenci z přikrytého hrnce utíkat. Měl by "vědět"to, že přetékající

kapalina může uhasit plamen vařiče, aniž by ovlivnila funkci přívodu plynu do hořáku. Bylo by víc než nešťastné, kdyby v takovém případě uvažovaný robot nebyl vybaven schopností předvídat vzrůst koncentrace plynu v místnosti a následně nebyl schopen vyřešit druhy nebezpečí z toho plynoucích. [30]

Báze dat

K řešení různých problémů je potřeba systému poskytnout data. Tato data se ukládají do báze dat a v případě potřeby se dosadí do formulací znalostí při vyhodnocování závěru. Konkrétní data např. může poskytovat uživatel systému skrze uživatelské rozhraní počítače. Vniká tak „dialog“, který simuluje rozhovor mezi expertem a nezkušenou osobou. Úlohou ES v tomto dialogu je dotazovat se na informace k dané problematice, analyzovat je a na základě znalostí (z báze znalostí) vytvořit závěr, nebo navrhnout řešení.

Interferenční (řídící) mechanismus

Řídicí mechanismus je programový modul a komponenta ES, která vyhodnocuje a zprostředkovává komunikaci mezi bází znalostí a bází dat. Příkladem části mechanismu může být např. jednoduché pravidlo typu „IF (vlhkost_nad_65 OR nyni_prsi) THEN (zapni_odvlhcovac AND zavri_okna)“.

6.2 Strojové učení

Strojové učení je podoblastí umělé inteligence, zabývající se algoritmy a technikami, které umožňují počítačovému systému 'učit se'. Učením v daném kontextu rozumíme takovou změnu vnitřního stavu systému, která zefektivní schopnost přizpůsobení se změnám okolního prostředí. [31]

Strojové učení se značně prolíná s oblastmi statistiky a dobývání znalostí (také označované jako dolování z dat) a má široké uplatnění. Jeho techniky se využívají např. v biomedicínské informatice (tzv. systémy pro podporu rozhodování), rozlišení nelegálního užití kreditních karet, rozpoznávání řeči a psaného textu a v mnoha dalších odvětvích. [31]

Strojové učení lze rozdělit podle způsobu učení do následujících kategorií:

- **Učení s učitelem (en: supervised learning)** – pro vstupní data je určen správný výstup (třída pro klasifikaci nebo hodnota pro regresi) .
- **Učení bez učitele (en: unsupervised learning)** – ke vstupním datům není znám výstup .
- **Kombinace učení s učitelem a bez učitele (en: semi-supervised learning)** – část vstupních dat je se známým výstupem, ale další data, typicky větší, jsou bez něj. Často se používá algoritmus očekávané hodnoty-maximalizace(en: Expectation–Maximization algorithm)
- **Zpětnovazebné učení (en: reinforcement learning)**, též učení posilováním.

Strojové učení je velmi rozšířené a zahrnuje pod sebe mnoho podoblastí např. rozhodovací stromy, Bayesovské sítě nebo neuronové sítě. Právě poslední zmiňované neuronové sítě se velmi často využívají např. pro vyhodnocování obrazu.

6.2.1 Hluboké učení (deep learning)

Hluboké učení je disciplína v rámci strojového učení, která se zabývá využitím algoritmů (především neuronových sítí) s velkým počtem vrstev (layers) reprezentujících data. Tato moderní technika je inspirována neuronovou sítí člověka. Je založena na velkém množství stejných fixně svázaných výpočetních jednotek, neuronů, které přecházejí do různých stavů na základě vážených vstupů.

Tyto stavy (parametry sítě) se pak dají natrénovat (naučit) a to pomocí učitele, bez učitele nebo s částečným dohledem učitele. Nejvyužívanější je však metoda učení s učitelem, která je popsána v dalším odstavci.

„Pokud je cílem vytvořit systém, který umožňuje určit, zda na obrázku je dům, auto nebo člověk, je nejprve potřeba připravit velkou tzv. trénovací množinu dat, která obsahuje obrázky domů, aut a lidí, kde každý jednotlivý obrázek obsahuje informaci, do které z kategorií (tříd) objekt spadá. Avšak taková množina dat s popisy se často připravuje ručně nebo jen částečně automaticky, což je hlavní nevýhoda učení s učitelem. Během tréninku pak systém prochází obrázky a vytvoří výstup ve formě vektoru hodnocení pro každou kategorii (třídu). Cílem je, aby všechny existující třídy měly co nejvyšší tzv. ohodnocující skóre, ale takový výsledek je bez trénování nepravděpodobný. Během procesu učení se proto počítá funkce, která měří chybu mezi hodnocením výstupu a vzorovým hodnocením. Následně systém upraví své nastavení s cílem snížení této chyby. Po trénování se výkon nastaveného systému ověří na jiném datovém balíku, který se nazývá testovací množina. Ta slouží k otestování obecných schopností nastaveného systému a určení výkonu na datech, se kterými se systém ještě nesetkal.“ [31]

Hluboké učení je v této době velmi rozšířená oblast AI, aniž by si to běžná populace uvědomovala. Pomocí hlubokého učení se totiž řídí sociální média, nebo se zlepšuje kvalita vyhledávání v prohlížečích. Je možné, že za chvíli bude monitorovat i naše životně důležité orgány. Není proto udivující, že se tato forma AI pomalu integruje i do domácích zařízení, spotřebičů a společně za pomoci dalších senzorů se zapojuje do chodu budov.

Další příklad uplatnění této technologie byl popsán v prezentaci Václava Matouška. [31] Výzkumníci z Massachusettského technologického institutu (MIT) vyvinuli systém, který umožní integraci neuronových sítí hlubokého učení s novými a mnohem menšími zařízeními, jako jsou drobné počítačové čipy v nositelných lékařských zařízeních, domácích spotřebičích a 250 miliardách dalších zařízení, která společně tvoří internet věcí. Díky takové kombinaci technologií (IoT a AI) by rozhodnutí o běžných domácích činnostech (např. kdy prát, kdy a jak mýt nádobí atp.), mohla řešit právě umělá inteligence, jež by se učila z našich návyků a zpětné vazby. Co ale dále stojí za zmínku,

je uplatnění hlubokého učení ve zdravotnictví. To by se mohlo spoléhat nejen na doktory, ale i na závěry umělé inteligence, která by byla schopna detekovat např. riziko kardiovaskulárních chorob.

6.3 Důvody využití AI/ML pro řízení provozu budovy

Umělá inteligence je společně s technologií internetu věcí postupně integrována do nových budov a domů, aby zkvalitnila podmínky pro práci či bydlení obyvatel, snížila energetické výdaje a zefektivnila využití prostoru.

Na základě analýz je umělá inteligence schopna samostatně integrovat a šířit data z chytrých (IoT) zařízení, učit se z nich, optimalizovat výkon a zlepšit efektivitu prostředí. Velká řada informací z digitálních zařízení poskytuje vnitřní pohled na operace, použití a podmínky infrastruktury budovy, vzduchu, vody, spotřeby energie a mnoho dalšího pro zajištění příjemného prostředí pro obyvatele. AI by se mohla časem přizpůsobovat konkrétním podmínkám a potřebám díky senzorům a zpětné vazbě obyvatel budovy.

Souhrn hlavních důvodů pro využívání AI:

- Snížení výdajů na energie budovy.
- Zkvalitnění podmínek pro bydlení.
- Zefektivnění využití prostoru v budově.

6.4 Návrh řešení AI / ML jeho realizace a vyhodnocení

Pro řízení budovy může být umělá inteligence v mém systému vytvořena dvěma způsoby. První způsob by budovu řídil na základě hodnot z monitorovacích systémů a importu dat předpovědi počasí s alespoň denním předstihem. Druhý způsob by podle hodnot z monitorovacích systémů byl schopen počasí předpovědět a podle této předpovědi budovu řídit. První varianta řízení bude dále popsána s návrhem expertního systému pro řízení. Druhá možnost s vlastní předpovědí počasí je velmi náročná a dále popíši jen možný příklad užití s využitím strojového učení.

6.4.1 Řízení budovy pomocí AI s využitím vlastních informací

Předpovídat počasí není jednoduché, jelikož je velmi proměnlivé a závisí na mnoha faktorech a hodnotách. I přes tuto proměnlivost však lze v počasí nalézt určité vzory chování a řídit podle těchto vzorů i budovu. Po konzultaci s lidským expertem na počasí a dat ze systému pro monitoring byly tyto vzory sepsány a mohly by být dále použity k vytvoření jednoduchého rozhodovacího systému pro řízení budovy. Z vytvořeného systému pro záznam dat (kapitola 3), jsou k dispozici základní hodnoty pro předpověď počasí. Jedná se o vnitřní teplotu a vlhkost budovy, venkovní hodnoty tlaku, vlhkosti, teploty, importovaná data z meteorologické stanice ve městě (teplota, vlhkost, tlak, dále

může být importován také např. směr a rychlost větru) a informace o čase, ze kterého se dá např. vyhodnotit roční období.

Všechna roční období vykazují podobné specifické vzory proměnlivosti. Každé období ovlivňují různé povětrnostní vzory, na které se váží atmosférické fronty (studené, teplé a okluzní). Tyto fronty s sebou mohou přinést srážky společně s ochlazením, ale také teplé počasí s jasnou oblohou. Příchod těchto front lze primárně vyhodnotit pomocí sledování změn tlaku v atmosféře nebo směrem větru.

Před příchodem studené fronty většinou stoupá teplota, tlak může zvolna klesat a fouká většinou vítr z jižních směrů. Během studené fronty klesne teplota a můžou se vyskytnout srážky. Po přechodu studené fronty lze pozorovat vzestupnou tlakovou tendenci, teplota může mírně vzrůst a relativní vlhkost může být díky chladnějšímu vzduchu vyšší. Po přechodu studené fronty se směr větru majoritně změní na západní/severozápadní. Pokud např. v Praze dojde k ochlazení a fouká západní/severozápadní vítr, lze očekávat, že tento vítr dojde také do Ostravy přibližně za 5 hodin při rychlosti cca 50 km/h. Pro teplou frontu platí podobné chování (pokud uvažujeme zhoršení počasí), ale místo ochlazení přináší počasí teplejší kvůli vanutí teplých jižních větrů. Tlak ovšem za teplou frontou může dál mírně klesat nebo zůstává setrvalý. Za další vzor chování lze také považovat skutečnost, že pokud teplota v noci výrazně neklesne (vzhledem k denním teplotám) a vane teplý vítr z jižních směrů, lze další den očekávat příchod studené fronty.

Tyto vzory by se daly využít pro tvorbu expertního systému k vyhodnocování počasí, ale pravděpodobně by vytvořený systém nebyl moc spolehlivý, kvůli proměnlivosti počasí a nedostatečnému množství měřených dat. K vyhodnocování předpovědi počasí jsou speciálně navržené systémy, které potřebují obrovský výpočetní výkon, aby dokázaly toto velké množství dat zpracovat. K vyhodnocování počasí se aktuálně také používá i superpočítač v Ostravě v centru IT4Inovations. I přes masivní výpočetní výkon však nelze stoprocentně určit vývoj počasí. Dále se tedy budu zabývat řízením budovy s importem dat předpovědi počasí.

Pokud by byl pro systém budovy zajištěn dostatečný výpočetní výkon, mohly by být vzory počasí také nalezeny procesem dolování z dat (en: data mining) pomocí strojového učení. Tento proces bude více popsán v této kapitole v sekci 6.4.3.

6.4.2 Řízení budovy pomocí AI s vlastními a importovanými informacemi

Pro zefektivnění podmínek pro bydlení a snížení nákladů pro provoz budovy může být umělou inteligencí řízeno několik zařízení. Většina spotřebičů a zařízení je v domě k zajištění tepelného komfortu. V zimě musí být např. do radiátorů nebo podlahového topení přivedena teplá voda pro zajištění dobrých podmínek pro bydlení. V kuchyni je po celý rok pro přípravu jídla využíván např. elektrický sporák. Všechna tato domácí zařízení spotřebovávají elektrickou energii. Pro úsporu této energie lze tato zařízení ovládat pomocí umělé inteligence, která by zajistila přísun jen takového množství energie, jež by bylo potřeba. Pro budovu takto lze vyhodnotit znalostní vzory, na jejichž

základě je možné společně s daty z monitorovacích systémů, vytvořit expertní systém pro řízení budovy.

Pro zajištění ideálních podmínek pro bydlení v budově lze vytvořit v expertním systému interferenční mechanismus, který bude řídit např. teplotu a vlhkost. Jako báze dat do tohoto mechanismu budou vstupovat informace o aktuálních hodnotách uvnitř domu a předpověď počasí na dnešní i další den. Jako báze znalostí budou v návrhu systému vystupovat znalosti pro řízení domu, např. vlhkost v domě během zimy je ideální, pokud se pohybuje v rozmezí 45 % až 60 %. Tento typ expertního systému je označován za tzv. pravidlový, protože obsahuje produkční pravidla, podle kterých se vyhodnocují další závěry. Základní strategie procesu usuzování závěrů by byla řízena daty a jednalo by se tak o dopředné řetězení (en: forward chaining). Postup této strategie by začal vyhodnocováním dostupných dat a znalostí směrem k závěru.

Příklad báze dat:

- Aktuálně je 20 °C.
- Za hodinu bude pravděpodobně 21 °C.
- Za hodinu bude pravděpodobně pršet.
- Akumulační jednotka má v sobě vodu o teplotě 60 °C.
- Maximální teplota akumulační jednotky je 90 °C.

Příklad báze znalostí:

- Pokud se zvyšuje tlak, nastane změna počasí.
- Pokud bude za hodinu pršet, zvýší se tlak.
- Vlhkost je v domě nízká, když klesne pod 35 %.
- Teplota v domě je ideální, když je v rozmezí 20 °C a 22 °C.
- Akumulační jednotka má dostatek energie na zítra, pokud její teplota vody dosahuje jejího maxima.

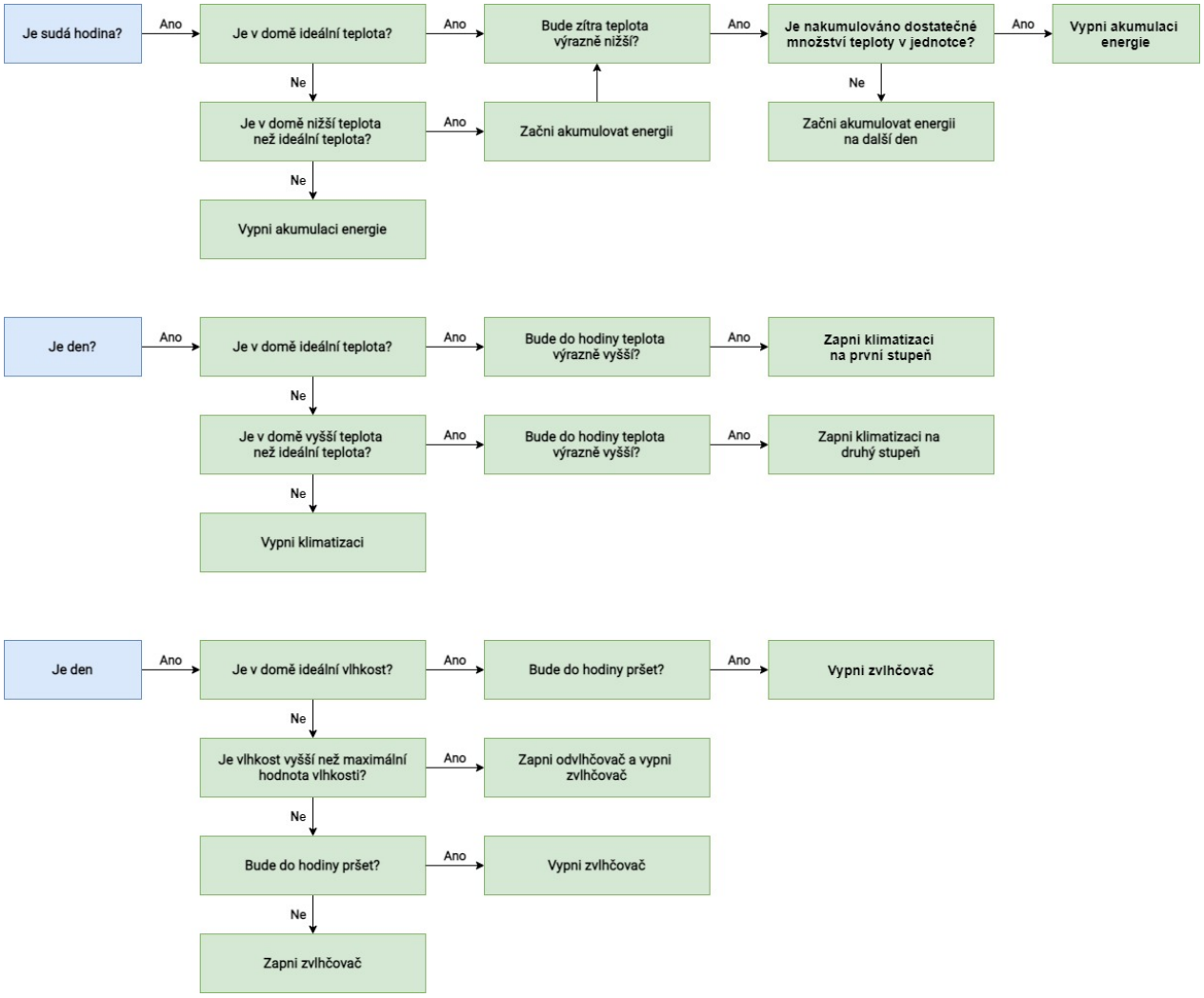
Příklad interferenčního mechanismu

Řídící mechanismus je postaven na informacích z báze znalostí a báze dat, podle kterých vyhodnocuje závěry pro řízení budovy. V odstavcích níže budou uvedeny v příkladech možnosti pro vytvoření funkčních jednotek v řídícím mechanismu. Na obrázku 6.2 je pak zobrazen rozhodovacího postup pro toto příkladné řízení.

První příklad využití tohoto mechanismu lze aplikovat na jednotce akumulační nádoby tepelného čerpadla. Tato nádoba slouží k ukládání teplé vody pro vyhřívání vodního podlahového topení. Pokud systém zjistí, že v budově není dostatečně teplo, pošle do trubek topení teplejší vodu, aby se teplota v domě dostala na ideální úroveň. Zároveň si systém přebere informaci, jestli má být zítra chladné počasí. Pokud ano, začne zvyšovat teplotu v akumulační nádobě, aby další den byla

k dispozici teplá voda pro ohřev topení. Tento příklad by byl vyhodnocován každé dvě hodiny z důvodu setrvačnosti teplot.

Druhý příklad lze podobným způsobem jako první aplikovat na ochlazení v budově. Pokud systém zjistí, že došlo k převýšení teploty v budově, zajistí spuštění chladícího zařízení k dosažení optimální teploty. Systém také zjistí, jestli má za další hodinu přijít větší oteplení a zajistí tak chlazení na vyšší stupeň, aby systém chladil budovu intenzivněji, než teplo z venku budovu zahřívalo. Tento příklad by byl vyhodnocován každou hodinu během dne. V noci by zvuk z chladícího zařízení rušil spánek obyvatel domu.



Obrázek 6.2: Příklad rozhodovacího postupu pro expertní systém

Třetí příklad by mohl vyrovnávat vlhkost v budově. Udržování ideální vlhkosti v budově je velmi důležité kvůli zdraví obyvatel budovy a ochrany před tvorbou plísně. Pokud systém zjistí, že došlo k převýšení nebo snížení vlhkosti, zapne podle toho zvlhčovač, nebo odvlhčovač. Pokud

je vlhkost nižší než minimální doporučená hodnota, před zapnutím zvlhčovače zjistí, jestli bude do hodiny pršet a pokud ano, tak zvlhčovač nezapne z důvodu ušetření energie. Tento příklad by byl vyhodnocován každou hodinu během dne.

Z obrázku 6.2 lze vidět, že je třeba definovat, kdy je akumulováno dostatečné množství teploty v jednotce nebo kdy nastává ideální teplota v domě. Toto jsou znalosti v bázi znalostí a jednotlivé hodnoty jako např. aktuální teplota v domě (např. 23 °C) jsou hodnoty z báze dat. Tento řídící mechanismus by mohl být naprogramován v jazyce Python (lokální server NAS Python podporuje) a spouštěn v požadovaných časech ve dne nebo v noci. Výsledkem by tak mohl být funkční systém pro řízení budovy, který by udržoval ideální vnitřní podmínky pro pohodlné bydlení.

6.4.3 Využití strojového učení k analýze dat a řízení budovy

V odstavci Řízení budovy s pomocí AI s využitím vlastních informací (6.4.1) byl zmíněn pojem dolování z dat. Tento proces umožňuje získat skryté a zajímavé informace a vzory ze zdrojových dat. Na velkém množství dat je spuštěn algoritmus pro dolování, který se tyto skryté vzory a informace snaží nalézt. Jedna z oblastí pro dolování z dat je oblast temporálních dat. Temporální data v sobě obsahují i časovou složku, která určuje jejich uspořádání a lze tak díky této vlastnosti najít periodické vzory, které popisují události vyskytující se v datech s určitou pravidelností. K této analýze a dolování se využívají také algoritmy strojového učení.

Dolování z dat a strojové učení jsou ale dvě odlišná odvětví datové vědy. Dolování z dat je proces extrakce užitečných informací z velkého množství dat. ML je na druhou stranu proces objevování algoritmů, které se učí na základě odvozených dat. Oba dva procesy se používají k řešení složitých problémů a lidé je občas mezi sebou zaměňují. K analýze procesu dolování se používá strojové učení, ale data z dolování se také využívají k učení strojů a proto jsou tyto dva pojmy zaměňovány.

Řízení budovy pomocí strojového učení by mohlo tedy probíhat dvěma způsoby. První způsob by pomocí algoritmů ML analyzoval historická data počasí (minimálně rok) a snažily by se hledat periodicity počasí během roku. Díky těmto periodicitám by mohl být s určitou pravděpodobností předpovězen vývoj současného počasí. Jednalo by se tedy o dolování z dat a následné učení dalších algoritmů pro předpověď počasí. Výsledkem by tak mohly být pravděpodobnostní hodnoty vývoje počasí tzn. například, s jakou pravděpodobností bude za hodinu pršet, nebo jaká bude pravděpodobně teplota další den. Systém by tak podle pravděpodobnostní hodnoty mohl rozhodovat o dalším řízení budovy a dále používat expertní systém navržený v odstavci Řízení budovy pomocí AI s vlastními a importovanými informacemi. 6.4.2

Kapitola 7

Shrnutí

V úvodu bakalářské práce bylo popsáno, jak lze v dnešní době číst, přenášet a ukládat data z okolního světa a jaká zařízení/platformy se k tomuto účelu využívají. Dále byla také popsána budova pro experimentální ověřování, ve které byl poté integrován ucelený systém pro monitoring provozu.

Ve druhé kapitole byly popsány technické platformy pro záznam a zpracování dat. V první části byla vyzkoušena zařízení (senzory) pro záznam dat a byly popsány jejich výhody a nevýhody. Různé senzory nabízí různou přesnost měření, a tak záleží, kde a v jakých podmínkách budou zařízení nasazena. Dále byla vybrána vhodná zařízení pro záznam a zpracování dat do systému budovy. V systému budovy byly jako zařízení pro záznam dat použity senzory DHT21 (záznam teploty a vlhkosti v budově), DS18B20 (záznam teploty trubek okruhů tepelného čerpadla) a BME280 (záznam teploty, vlhkosti a tlaku v okolí budovy). Tyto senzory byly použity z důvodu vyhovující spolehlivosti, přesnosti měření a dostupné ceně. Dále byla v této kapitole popsána zařízení pro zpracování dat ze senzorů. Jako hlavní zařízení pro zpracování dat bylo vybráno zařízení NodeMCU-S ESP32 z důvodu spolehlivého přenosu dat, možnosti využití mnoha důležitých funkcí pro chod a šetření energie (RTC, hluboký spánek, NTP, ...) a kompaktních rozměrů. Mikrokontroléry ESP-01, Arduino a Raspberry PI byly pro zpracování dat také otestovány, avšak nebyly v systému použity z důvodu nespolehlivého připojení (v případě ESP-01), velké spotřeby energie (v případě Arduina) nebo velké ceně (v případě Raspberry PI).

Ve třetí kapitole bylo navrženo řešení systému pro monitoring budovy obsahující tři podsystémy. První podsystém byl určen pro monitoring teploty a vlhkosti v budově. Druhý podsystém byl určen pro monitoring teploty pracovních kapalin tepelného čerpadla a třetí podsystém byl vytvořen pro monitoring teploty, vlhkosti a tlaku v okolí budovy. Pro všechny podsystémy byly přiřazeny senzory a mikrokontrolér NodeMCU-S ESP32 vybrané v druhé kapitole a byly mezi sebou fyzicky propojeny. Po rozvržení podsystémů, výběru jejich technické platformy a jejich správnému fyzickému zapojení byl vytvořen a popsán program pro záznam dat. Pro ukládání dat byl poté zprovozněn vlastní server popsaný ve čtvrté kapitole.

Čtvrtá kapitola se zabývala médii pro ukládání dat a možnostmi jejich síťového ukládání. Byly

popsány rozdíly, výhody a nevýhody mezi pronájmem síťového úložiště přes poskytovatele síťových služeb (cloudu) a zakoupením domácího síťového úložiště (NAS). Obě varianty ukládání dat byly v mém systému vyzkoušeny, ale jako hlavní řešení bylo zvoleno ukládání dat na domácí síťové úložiště (server Synology NAS DS712) z důvodu nevyužité kapacity tohoto zařízení v původním systému budovy. Tento NAS server byl zprovozněn a byl na něm dále instalován software web server, který fungoval jako rozhraní pro ukládání, sdílení a vizualizaci dat. Dále byl vytvořen a popsán program pro ukládání dat na tomto web serveru a byly popsány taktéž technologie k tomuto účelu vybrané (technologie AMP). Pro přenos dat mezi systémy byl zvolen formát JSON z důvodu přehlednosti a dobré rychlosti přenosu. Jako poslední věc bylo v této kapitole popsáno, jak funguje import dat z předpovědního portálu.

V páté kapitole byla vytvořena webová stránka pro vizualizaci dat odkudkoliv z internetu. Pro vizualizaci byla také vytvořena mobilní aplikace na platformu Android. Na obou platformách se zobrazují data aktuální i data dlouhodobá. Webová stránka je dostupná na adrese http://www.iotcontrol.synology.me:10936 a je plně hostována na zprovozněném domácím NAS serveru.

V šesté kapitole byla obecně popsána umělá inteligence a její odvětví. Důraz byl kladen na expertní systémy a strojové učení, jelikož tyto formy umělé inteligence byly použity při návrhu systému umělé inteligence pro řízení budovy. Dále byla popsána možnost řízení budovy umělou inteligencí s vlastní předpovědí počasí a možnost řízení budovy s importem dat z předpovědního portálu. Pro druhou možnost byl navržen expertní systém pro automatické řízení teploty a vlhkosti v domě v závislosti na aktuálním i předpovídaném počasí. Tento systém by po naprogramování měl být schopen udržovat ideální vnitřní podmínky pro pohodlné bydlení v budově. Pro předpovídání počasí byla také popsána možnost využití algoritmů strojového učení, které by využily data z monitorovacích systémů budovy, a to jak nasbíraných, tak aktuálních.

Kapitola 8

Závěr

Úkolem této bakalářské práce bylo navrhnout a připravit řešení monitoringu provozu moderních budov pro jejich další možné řízení pomocí informačních technologií, umělé inteligence a strojového učení. Po úvodním seznámení se zařízeními pro záznam a zpracování dat byl navržen ucelený systém pro monitoring teploty, tlaku a vlhkosti v experimentální budově i mimo ni. Tento systém je rozdělen na tři podsystémy pro monitoring v různých částech budovy. Pro záznam dat tohoto systému byly vybrány senzory a zařízení pro zpracování dat, které vydrží fungovat ve vnitřních i vnějších podmínkách budovy. Pro každý podsystém byl vytvořen program pro záznam a odesílání dat na web server. Tyto podsystémy spolehlivě fungují a úspěšně se daří přenášet data přes bezdrátovou technologii Wifi na web server. V budoucnu by mohl být v těchto programech vylepšen systém zpracování a odchytávání chybných měření nebo nepovedených pokusů při připojení na server.

Po vytvoření všech podsystémů monitoringu byl zprovozněn domácí NAS server, jako platforma pro ukládání a práci s daty. Po instalaci software web serveru na tento domácí NAS server bylo dosaženo spolehlivého ukládání dat do vytvořené databáze. Data se tak podařilo úspěšně ukládat, mohla být dále zobrazena na dalších zařízeních a mohou být použita k vyhodnocení systému umělé inteligence. Kromě dat z monitoringu jsou do systému díky rozšířeným možnostem domácího serveru importována i data z předpovědního portálu. Instalovaný web server umožňuje také hostování vlastních webových stránek, které byly úspěšně vytvořeny. Na této stránce jsou vizualizována data ze senzorů ve formě grafů. Tato webová stránka je dostupná z lokální sítě, ale i odkudkoliv z internetu. Uživatel budovy se tak kdekoliv může podívat na aktuální hodnoty z monitorovacích zařízení budovy. V budoucnu by systém mohl být rozšířen o přihlašovací rozhraní pro zlepšení bezpečnosti. Kromě webové stránky byla vytvořena funkční mobilní aplikace na Android, která také zajišťuje možnost nahlížet na data z monitorovacích zařízení.

Po vytvoření systému pro monitoring a vizualizaci zaznamenaných dat byl navržen expertní systém umělé inteligence, který by dokázal řídit domácí zařízení. Tento expertní systém by využíval data z monitorovacích systémů budovy a také dat z předpovědi počasí. Výsledkem zhotovení tohoto expertního systému by tak mohl být funkční systém pro řízení budovy, který by udržoval ideální

vnitřní podmínky pro pohodlné bydlení. Zároveň by také poskytl dobrou informovanost obyvatelům budovy díky monitorovacím systémům. Kromě tohoto systému byla také popsána možnost řízení budovy na základě vlastní předpovědi počasí. Tato možnost řízení ale nebyla realizována z důvodu komplexnosti a proměnlivosti počasí. V budoucnu by tato problematika mohla být zpracována nejen expertním systémem, ale také strojovým učením a procesem dolování z dat, které bylo v poslední kapitole nastíněno.

Výstupem této bakalářské práce tedy není jen monitorovací systém pro budovu, ale také možný přístup automatického řízení prvků budovy. Řešení umělé inteligence, jež je v práci uvedeno, by mohlo přinést inspiraci pro automatické řízení budovy, snížení jejích provozních nákladů a zkvalitnění podmínek pro bydlení obyvatel.

Literatura

1. BOSIO, Manuel. *Internet of Things* [online]. [B.r.]. Dostupné také z: https://en.wikipedia.org/wiki/Internet_of_things?oldid=808022410.
2. Tecomat Foxtrot. *Teco Info*. 2018-03, s. 1–4.
3. *Metodické vysvětlivky - definice vybraných ukazatelů bytové výstavby* [online]. [B.r.]. Dostupné také z: https://www.czso.cz/csu/xb/metodicke_vysvetlivky_definice_vybranych_ukazatelu_bytove_vystavby.
4. ADRYAN, Boris; OBERMAIER, Dominik; FREMANTLE, Paul. *The technical foundations of IoT*. Boston: Artech House, [2017]. Mobile communications series. ISBN 978-1-63081-251-5.
5. *Strukturovaná kabeláž* [online]. [B.r.]. Dostupné také z: https://www.elkov.cz/sluzby-poradenstvi-a-navrhy-strukturovana-kabelaz/.
6. *SOLARIX kabel, CAT5E, UTP PVC, 305m, box* [online]. [B.r.]. Dostupné také z: https://www.tsbohemia.cz/solarix-kabel-cat5e-utp-pvc-305m-box_d125025.html.
7. *Teplotní čidlo a vlhkoměr DHT21/AM2301* [online]. [B.r.]. Dostupné také z: https://www.hadex.cz/m441b-teplotni-cidlo-a-vlhkomer-dht21am2301/.
8. VYLEGALA, Ing. Pavel. *Rozdělení snímačů* [online]. 2014. Dostupné také z: https://www.sse-najizdarne.cz/projekty/roboti/dokumenty/v_prez_ss_1.pdf.
9. HORÁK, Bohumil. *Senzory. Sylab předmětu ISTE* [online]. Ostrava, 2019.
10. *Drive-System Europe lineární servomotor* [online]. [B.r.]. Dostupné také z: https://www.conrad.cz/p/drive-system-europe-linearni-servomotor-dszy1-12-10-050-pot-ip65-1386431-delka-50-mm-1-ks-1386431.
11. *ATmega32* [online]. [B.r.]. Dostupné také z: https://www.microchip.com/wwwproducts/en/ATMEGA32.
12. *VPN router Cisco RV160W WLAN AC* [online]. [B.r.]. Dostupné také z: https://www.bechtle.com/cz/shop/vpn-router-cisco-rv160w-wlan-ac--4403192-15--p.
13. *Arduino UNO REV3* [online]. [B.r.]. Dostupné také z: https://cdn.antratek.nl/media/product/b03/arduino-uno-rev3-a000066-0f4.jpg.

14. *Raspberry Pi 3 Model B+* [online]. [B.r.]. Dostupné také z: https://www.tsbohemia.cz/raspberry-pi-3-model-b-_d292238.html.

15. *Waveshare NodeMCU-32S ESP32* [online]. [B.r.]. Dostupné také z: https://rpishop.cz/iot/2605-waveshare-nodemcu-32s-esp32-wifi-bluetooth-vyvojova-deska.html.

16. *ESP32 block diagram* [online]. [B.r.]. Dostupné také z: https://www.electronics-lab.com/espressif-next-gen-esp32-s2-soc-family-goes-mass-production/esp32-block-diagram/.

17. *ESP8266 ESP-01 WIFI TCP/IP* [online]. [B.r.]. Dostupné také z: https://dratek.cz/arduino/911-internet-veci-je-tady-tcp-ip-wifi-esp8266-esp-01.html?gclid=CjwKCAjwmv-DBhAMEiwA7xYrd1f7CQpgXn0teXLCkuMD9sPAtiIPjgTF2zjl1A2JtN2ZCe145yCxoBoCEhQQAvD_BwE.

18. *ESP8266 block diagram* [online]. [B.r.]. Dostupné také z: https://www.researchgate.net/figure/Figure-10-ESP8266-block-diagram-with-its-main-internal-functional-blocks_fig6_304929498.

19. *Modul teploměru a vlhkoměru s DHT11* [online]. [B.r.]. Dostupné také z: https://www.gme.cz/modul-teplomeru-a-vlhkomeru-s-dht11.

20. *Digital temperature sensor DS18B20* [online]. [B.r.]. Dostupné také z: https://nettigo.eu/products/digital-temperature-sensor-ds18b20.

21. *Teploměr vodotěsný - sonda 2m DS18B20* [online]. [B.r.]. Dostupné také z: https://dratek.cz/arduino/2029-teplomer-vodotesny-sonda-2m-ds18b20.html?gclid=Cj0KCQjw6-SDBhCMARIsAGbI7Uh_g8FZWbYQTOiZd_H3rtjLVveqYZVOXtsU0vr59Oobby8OJI1BxhAaAno5EALw_wcB.

22. *Senzor tlaku, teploty a vlhkosti BME280* [online]. [B.r.]. Dostupné také z: https://www.laskarduino.cz/arduino-senzor-tlaku--teploty-a-vlhkosti-bme280/?gclid=CjwKCAjwmv-DBhAMEiwA7xYrd-z9D8SOO6hE4D-s6oXkDnmpFaXgr0wl5gu8m49mt90pITX1ouXeLBoCATgQAvD_BwE.

23. *Modul meteočidla s BMP180 GY-68* [online]. [B.r.]. Dostupné také z: https://www.gme.cz/modul-meteocidlo-bmp180.

24. *Insight Into ESP32 Sleep Modes and Their Power Consumption* [online]. [B.r.]. Dostupné také z: https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/.

25. *An overview of HTTP* [online]. [B.r.]. Dostupné také z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.

26. MIESSLER, Daniel. *What's the Difference Between a URI and a URL?* [Online]. [B.r.]. Dostupné také z: https://danielmiessler.com/study/difference-between-uri-url/.

27. VOLNÁ, Eva. *Umělá inteligence*. Ostrava, 2013.

28. PĚCHOUČEK, Michal; ROLLO, Milan. *Neinformované metody prohledávání stavového prostoru* [online]. Ostrava: CRC Press, Taylor a Francis Group, 2020. Dostupné také z: https://cw.fel.cvut.cz/old/_media/courses/a3b33kui/prednasky/prednaska_04.pdf?cache=nocache.

29. *Expertní systémy* [online]. [B.r.]. Dostupné také z: https://www.kiv.zcu.cz/studies/predmety/uzi/Literatura/Expertni_systemy3.pdf.

30. CELBOVÁ, Iva. *Úvod do problematiky expertních systémů* [online]. 1999. Dostupné také z: https://ikaros.cz/uvod-do-problematiky-expertnich-systemu.

31. *Strojové učení* [online]. 2020. Dostupné také z: https://www.kiv.zcu.cz/studies/predmety/uzi/Folie_ZS/Stroj_uceni.pdf.

Příloha A

Popis společnosti Teco a.s. a jejich chytré budovy

TECO a.s. je česká společnost se sídlem v Kolíně, která vznikla v roce 1993 a zabývá se výrobou průmyslových řídicích systémů kategorie PLC. Tyto systémy je možné nasadit jak do průmyslových, tak do běžných budov. Jejich produkty jsou aplikovatelné v nejrůznějších oblastech automatizace, řízení a ovládání. Lze je aplikovat např. k vytápění, chlazení, osvětlení, větrání, chytrého nabíjení, ovládání žaluzií a bran, řízení přístupu, měření teploty, měření ostatních veličin a k mnohým dalším věcem. Jeden z modulárních řídicích systémů má pojmenování Tecomat Foxtrot a je často využíván v inteligentních systémech díky výkonné procesorové jednotce s architekturou RISC a promyšlenému systému vstupně/výstupních periferií. Je také označován za řídicí systém nové generace. Tento řídící systém je také využit v chytré budově této firmy v Kolíně.

Chytrá budova Teco a.s.

Budova byla architektem rozdělena do čtyř částí. Tyto čtyři provozní části se dají oddělit a samostatně uzavřít. Stavba na sebe neupozorňuje výrazným barevným řešením. Je navržena v odstínech stříbřitě šedé a je zajímavá jednoduchými a čistým liniemi. Tyto linie prostupuji i do interiéru, kde se již barevnost loga firmy objevuje, a to ve formě modrého pruhu na podlaze vstupní haly. Interiér budovy oživují různobarevná řešení podlah v kancelářích. [2] Pásová okna s venkovními žaluziemi lemují celý obvod budovy a zajišťují tak přirozené osvětlení a větrání uvnitř budovy.

Dalším specifikem budovy je její energetický koncept, jelikož je zaměřen na maximální efektivitu hospodaření s energií. Kromě zateplení pláště budovy bylo zvoleno nízkoteplotní podlahové vytápění a stropní chlazení. Voda v topení je poháněna dvojící tepelných čerpadel od firmy PZP. Díky těmto čerpadlům je zajištěna ideální teplota v zimě i v létě. Dalším příkladem maximální efektivity budovy je využití odpadního tepla ze serverů a technologie pájení součástek k ohřevu topné vody v zimě.

Za veškerá měření a regulaci této topné a chladící soustavy je zodpovědný řídící systém Tecomat Foxtrot. [2]

Tato budova využívá k fungování výhradně elektřinu. Elektřina je zde použita na vytápění, chlazení, ventilaci, osvětlení atd. Budova zatím neobsahuje fotovoltaické panely, ale jejich instalace je připravena do další etapy jejího vývoje. Veškerá zařízení, zásuvky, vypínače, žaluzie a termostaty jsou připojeny na více než dvacítku větví instalační sběrnice CIB. Tecomat Foxtrot je zde také využíván pro fungování přístupového systému a zabezpečovací ústředny, do které je také připojen kamerový systém. Budova je připravena do budoucna na další změny v chování i provozu podle nejrůznějších kombinací logických podmínek a měřených dat, například spotřeb energií. [2]

Obrázek A.1: Chytrá budova společnosti Teco a.s., zdroj: http://tecoacademy.cz/

Příloha B

Programový kód podsystému pro záznam teplot trubek tepelného čerpadla

```
#include <WiFi.h> /* knihovna pro komunikaci mezi serverem a ESP32 */
#include <ArduinoJson.h> /* knihovna pro "parsing" formátu JSON */
#include "time.h" /* knihovna práci s časem */

#include <OneWire.h> /* knihovna pro komunikaic se sběrnicí OneWire */
#include <DallasTemperature.h> /* knihovna pro práci se senzorem DS18B20 */
#define ONE_WIRE_BUS 17 /* definice pinu pro datovou sběrnici */

OneWire oneWire(ONE_WIRE_BUS); /* nastavení komunikace vytvořením OneWire instance
    na konkrétním pinu */
DallasTemperature sensors(&oneWire); /* inicializace proměnné sensors, díky které
   budou dále čteny teploty ze všech DS18B20 senzorů */

#define uS_TO_S_FACTOR 1000000ULL /* konverzní faktor z mikrosekund na sekundy */
#define TIME_TO_SLEEP 10      /* čas po který bude ESP32 v režimu spánku v sekundá
   ch - 900s = 15min */

RTC_DATA_ATTR int bootCount = 0;

const char* ssid   = "jmeno_wifi";
const char* password = "heslo_wifi";

const char* host = "192.168.0.202";
```

```
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 3600;
const int  daylightOffset_sec = 3600;

void printLocalTime()
{
  /* funkce pro vrácení aktuálního času */
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    return;
  }
  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
}
String getActLocalTime()
{
  /* funkce pro vrácení aktuálního času ve formátu pro odeslání na server */
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
    return "";
  }

  String myDate;
  String myYear = String(timeinfo.tm_year + 1900);
  String myMonth = String(timeinfo.tm_mon + 1);
  String myDay = String(timeinfo.tm_mday);
  String myHour = String(timeinfo.tm_hour);
  String myMin = String(timeinfo.tm_min);
  String mySec = String(timeinfo.tm_sec);

  myDate = myYear;

  myDate += (myMonth.length() < 2) ? ("-0" + myMonth) : ("-" + myMonth);
  myDate += (myDay.length() < 2) ? ("-0" + myDay) : ("-" + myDay);
  myDate += "T";
  myDate += (myHour.length() < 2) ? ("0" + myHour) : (myHour);
```

```
  myDate += (myMin.length() < 2) ? (":0" + myMin) : (":" + myMin);
  myDate += (mySec.length() < 2) ? (":0" + mySec) : (":" + mySec);
  //myDate += "Z";

  return myDate;
}
void print_wakeup_reason(){
  /* funkce pro vypsání důvodu probuzení */
  esp_sleep_wakeup_cause_t wakeup_reason;
  wakeup_reason = esp_sleep_get_wakeup_cause();

  switch(wakeup_reason)
  {
    case ESP_SLEEP_WAKEUP_EXT0 : Serial.println("Wakeup caused by external signal
        using RTC_IO"); break;
    case ESP_SLEEP_WAKEUP_EXT1 : Serial.println("Wakeup caused by external signal
        using RTC_CNTL"); break;
    case ESP_SLEEP_WAKEUP_TIMER : Serial.println("Wakeup caused by timer"); break;
    default : Serial.printf("Wakeup was not caused by deep sleep: %d\n",
        wakeup_reason); break;
  }
}
void goToSleep(){
  /* funkce pro ukládání zařízení ke spánku */
  Serial.println("Going to sleep now for " + String(TIME_TO_SLEEP) + " Seconds");
  Serial.flush();
  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
  esp_deep_sleep_start(); /* kód za touto funkcí se nidky neprovede */
}

void printWifiStats(){
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void setup(){
```

```
Serial.begin(9600);

bootCount++;
Serial.println("Boot number: " + String(bootCount));
print_wakeup_reason();

Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid, password);

unsigned long timeout = millis();

while (WiFi.status() != WL_CONNECTED) {
  if(millis() - timeout > 5000){
    Serial.println("Connection to wifi failed!");
    //digitalWrite(LED,HIGH);
    goToSleep();
  }
  else{
    delay(500);
    Serial.print(".");
  }
}

//printWifiStats();

configTime(gmtOffset_sec, daylightOffset_sec, ntpServer); //inicializace času
String myTime = getActLocalTime();

if(myTime == ""){
   Serial.println(F("Failed to read time!"));
   goToSleep();
}
printLocalTime();
```

```
float temps[4];
sensors.begin(); /* inicializace senzorů na sběrnici OneWire */
delay(50);
sensors.requestTemperatures();

for (int i = 0; i < 4; i++)
{
  Serial.print("Sensor ");
  Serial.print(i);
  Serial.print(" : ");
  temps[i] = sensors.getTempCByIndex(i); /* čtení teploty ze senzoru */
  Serial.print(temps[i]);
  Serial.print("C | ");
  if (isnan(temps[i])) {
    Serial.println("Failed to read from DS18B20 sensor.");
    goToSleep();
  }
}
Serial.println("");

WiFiClient client;
const int httpPort = 80;
if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    goToSleep();
}

String url = "/php/pump/insert.php?";
String table = "Heat_pump";

url = url + "inner_in=" + temps[0] + "&";
url = url + "inner_out=" + temps[1] + "&";
url = url + "outer_in=" + temps[2] + "&";
url = url + "outer_out=" + temps[3] + "&";
url = url + "dayTime=" + myTime + "&";
url = url + "table=" + table;

/* Odeslání dat na server pomocí HTTP Get požadavku */
```

```
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Authorization: Basic amVycnk6MTIzNA==" + "\r\n" +
               "Connection: keep-alive\r\n\r\n");

  Serial.println(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Authorization: Basic amVycnk6MTIzNA==" + "\r\n" +
               "Connection: keep-alive\r\n\r\n");

  timeout = millis();

  while (client.available() == 0) {
      if (millis() - timeout > 5000) {
          Serial.println(">>> Client Timeout !");
          client.stop();
          goToSleep();
      }
  }
  while(client.available()) {
      String line = client.readStringUntil('\r');
      Serial.print(line);
  }
  Serial.println();
  Serial.println("closing connection");

  Serial.println("");
  Serial.println("Everything went perfectly!");
  goToSleep();
}

void loop(){
  /* Tato funkce nebude nikdy volána, protože smyčka programu se vytváří opakovaný
      m probouzením ESP32 */
}
```

Příloha C

Programový kód podsystému pro záznam teploty, vlhkosti a tlaku v okolí budovy

```
#include <WiFi.h> /* knihovna pro komunikaci mezi serverem a ESP32 */
#include <ArduinoJson.h> /* knihovna pro "parsing" formátu JSON */
#include "time.h" /* knihovna práci s časem */

#include <Wire.h> /* knihovna pro komunikaic s I2C */
#include <Adafruit_Sensor.h> /* Adafrit knihovna pro práci se senzory */
#include <Adafruit_BME280.h> /* Adafrit knihovna pro práci s BME280 */

#define uS_TO_S_FACTOR 1000000ULL /* konverzní faktor z mikrosekund na sekundy */
#define TIME_TO_SLEEP 900      /* Čas po který bude ESP32 v režimu spánku v sekund
    ách - 900s = 15min */

RTC_DATA_ATTR int bootCount = 0;

const char* ssid   = "jmeno_wifi";
const char* password = "heslo_wifi";

const char* host = "192.168.0.202";

const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 3600;
const int  daylightOffset_sec = 3600;

Adafruit_BME280 bme; // instance pro komunikaci přes I2C
```

```
void printLocalTime()
{
  /* funkce pro vrácení aktuálního času */
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    return;
  }
  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
}
String getActLocalTime()
{
  /* funkce pro vrácení aktuálního času ve formátu pro odeslání na server */
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
    return "";
  }

  String myDate;
  String myYear = String(timeinfo.tm_year + 1900);
  String myMonth = String(timeinfo.tm_mon + 1);
  String myDay = String(timeinfo.tm_mday);
  String myHour = String(timeinfo.tm_hour);
  String myMin = String(timeinfo.tm_min);
  String mySec = String(timeinfo.tm_sec);

  myDate = myYear;

  myDate += (myMonth.length() < 2) ? ("-0" + myMonth) : ("-" + myMonth);
  myDate += (myDay.length() < 2) ? ("-0" + myDay) : ("-" + myDay);
  myDate += "T";
  myDate += (myHour.length() < 2) ? ("0" + myHour) : (myHour);
  myDate += (myMin.length() < 2) ? (":0" + myMin) : (":" + myMin);
  myDate += (mySec.length() < 2) ? (":0" + mySec) : (":" + mySec);
  //myDate += "Z";
```

```
  return myDate;
}
void print_wakeup_reason(){
  /* funkce pro vypsání důvodu probuzení */
  esp_sleep_wakeup_cause_t wakeup_reason;
  wakeup_reason = esp_sleep_get_wakeup_cause();

  switch(wakeup_reason)
  {
    case ESP_SLEEP_WAKEUP_EXT0 : Serial.println("Wakeup caused by external signal
        using RTC_IO"); break;
    case ESP_SLEEP_WAKEUP_EXT1 : Serial.println("Wakeup caused by external signal
        using RTC_CNTL"); break;
    case ESP_SLEEP_WAKEUP_TIMER : Serial.println("Wakeup caused by timer"); break;
    default : Serial.printf("Wakeup was not caused by deep sleep: %d\n",
        wakeup_reason); break;
  }
}
void goToSleep(){
  /* funkce pro ukládání zařízení ke spánku */
  Serial.println("Going to sleep now for " + String(TIME_TO_SLEEP) + " Seconds");
  Serial.flush();
  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
  esp_deep_sleep_start(); /* kód za touto funkcí se nidky neprovede */
}

void printWifiStats(){
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void setup(){
  Serial.begin(9600);

  bootCount++;
```

```
Serial.println("Boot number: " + String(bootCount));
print_wakeup_reason();

Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid, password);

unsigned long timeout = millis();

while (WiFi.status() != WL_CONNECTED) {
  if(millis() - timeout > 5000){
    Serial.println("Connection to wifi failed!");
    goToSleep();
  }
  else{
    delay(500);
    Serial.print(".");
  }
}

//printWifiStats();

configTime(gmtOffset_sec, daylightOffset_sec, ntpServer); //inicializace času
String myTime = getActLocalTime();

if(myTime == ""){
   Serial.println(F("Failed to read time!"));
   goToSleep();
}
printLocalTime();

bool status = bme.begin(0x76); /* inicializace senzoru */
if (!status) {
  Serial.println("Could not find a valid BME280 sensor, check wiring!");
  goToSleep();
```

```
}
float t = bme.readTemperature(); /* čtení veličin ze senzoru */
float h = bme.readHumidity();
float p = bme.readPressure() / 100.0F; /* převedení tlaku na hPa */

if (isnan(h) || isnan(t) || isnan(p)) {
  Serial.println(F("Failed to read from BME280 sensor!"));
  goToSleep();
}

WiFiClient client;
const int httpPort = 80;
if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    goToSleep();
}

String url = "/php/outdoor/insert.php?";
String table = "Outdoor_temphumpress";

url = url + "temperature=" + t + "&";
url = url + "humidity=" + h + "&";
url = url + "pressure=" + p + "&";
url = url + "dayTime=" + myTime + "&";
url = url + "table=" + table;

/* Odeslání dat na server pomocí HTTP Get požadavku */
client.print(String("GET ") + url + " HTTP/1.1\r\n" +
             "Host: " + host + "\r\n" +
             "Authorization: Basic amVycnk6MTIzNA==" + "\r\n" +
             "Connection: keep-alive\r\n\r\n");

Serial.println(String("GET ") + url + " HTTP/1.1\r\n" +
             "Host: " + host + "\r\n" +
             "Authorization: Basic amVycnk6MTIzNA==" + "\r\n" +
             "Connection: keep-alive\r\n\r\n");

timeout = millis();
```

```
  while (client.available() == 0) {
      if (millis() - timeout > 5000) {
          Serial.println(">>> Client Timeout !");
          client.stop();
          goToSleep();
      }
  }
  while(client.available()) {
      String line = client.readStringUntil('\r');
      Serial.print(line);
  }
  Serial.println();
  Serial.println("closing connection");

  Serial.println("");
  Serial.println("Everything went perfectly!");
  goToSleep();
}

void loop(){
  /* tato funkce nebude nikdy volána, protože smyčka programu se vytváří opakovaný
      m probouzením ESP32 */
}
```

Příloha D

Fotografie podsystému pro záznam teplot trubek tepelného čerpadla